Detection of Unsolicited Web Browsing with Clustering and Statistical Analysis

by

Pawel Chwalinski

Thesis

Submitted to the University of Middlesex

for the degree of

Doctor of Philosophy

School of Science and Technology

September 2013

Contents

List of Tables

List of Figures

Abbreviations

Attacks - There are two main attack trends focused at in this thesis:

DoS - DoS stands for Denial of Service, and it constitutes popular computer attack against legitimate targets. The result of the attack is some form of preventions of the legitimate users from accessing a network service.

DDoS - DDoS stands for Distributed Denial of Service. The attack is spread across a number of machines. Often, the volume of traffic generated by the attacking hosts is large enough to cause services unresponsive.

RDDoS - RDDoS stands for Reflector DDoS. This attack is modification of standard DDoS where network communication is not essentially established between an attacking host and a target. On the contrary, the RDDoS attack takes place when a number of "reflectors" send packets to other legitimate machines, called reflectors, and inserts the source address as the address of the target. As a result, the reflectors establish communication channel with the target, and attempt to resolve the lost communication between each other, that has never taken place.

Detailed description of the attacks has been complied in Section 2.1.

Datasets - Throughout this thesis the following four datasets are considered:

CLARKENT - CLARKNET dataset spans two weeks (in August-September, 1995) of activity, and contains 3,328,587 requests.

ESHOP - ESHOP dataset has been kindly provided by Giralte et al. [28]. Inside this web-log there are only 64,044 different sessions observed during one

day in 2009. The data set has been obtained to validate the proposed approach for a relatively *smaller* and *newer* data set.

MSNBC - MSNBC dataset has been obtained during a day-long activity of a web server, on the 28^{th} of September, 1999. It consists of 989,818 sequences of requests.

NASA - NASA dataset has been recorded in two weeks in (in July-August, 1995), and contains 3,461,612 requests.

Packets - Typical three way handshake during TCP/IP connection establishment consists of the following packets:

SYN - Initially, a first step of the handshake procedure begins with a SYN synchronisation packet. Clients generates its own sequence number $seq_c = x$, used for the order control of transmitting packets.

SYN, ACK - The availability of the target service and confirmation of the SYN packet arrival is carried out by transmission of SYN, ACK packet. The target machine replies with its own sequence number $seq_t = y$, and acknowledgement number of the client first packet $ack_t = x + 1$

ACK - Client confirms the handshake completion by incrementing its own acknowledgement number $ack_c = y + 1$

IDS Performance - The performance of IDS's is measured by analysis of the following parameters

True Positive - is an event when actual illegitimate activity is correctly recognised

False Negative - denotes an event when malicious activity is marked as legitimate attempt

True Negative - is an event when legitimate activity is marked correctly as harmless attempt

False Positive - false positive alarm describes misclassification of legitimate activity as malicious

ROC Curve - In addition, the performance of IDS can be represented with Receiver Operating Characteristic (ROC) curve that provides insight into overall classifier's accuracy. This curves depicts a trade-off between FP and TP.

Notations

Dataset the following notation is used to described datasets and what they consist of

D - denotes a dataset. Each dataset consists of ordered requests generated from an IP address, as they appeared during dataset recording. Detailed description can be found in Section 3.1.2. Obtained datasets are split into two datasets:

T - a subset of **D** dataset for training purposes

V - a subset of **D** dataset for validating patterns obtained having analysed **T**.

B_i - in some cases, **T** is split into partitions $B_i, 1 \leq i \leq p$ to improve computational speed of the resource-intensive algorithm.

A - In addition, there are datasets with manually generated sequences of requests for a given website, and denoted with **A**. These are further categorised into the following various browsing strategies:

A_{C_R} - denotes hosts Rarely-Changing categories and is fully described in Section 4.9.1.

A_{C_F} - denotes hosts Frequently-Changing categories and is fully described in Section 4.9.2.

A_H - denotes a datasets consisting of requests generated by hosts that browse website similarly to human agents, and is described Section 4.10.

A_N - denotes a set of strategies generated by naive attackers, and is fully described in Section 4.10.

A_R - denotes a dataset of hosts browsing a website in a random-walk fashion and is described in Section 4.10.

s_i - denotes a sequence of i-th user inside a dataset.

$r_\wedge$ - denotes a minimum number of requests observed in a dataset for a user.

$r_\vee$ - denotes a maximum number of requests observed in a dataset for a user.

C_j - denotes j-th cluster, and $1 \leq j \leq k$.

Ω - all unique requests observed in $\mathbf{D}$, are stored inside set Ω. In a particular case, set Ω is extended by one element (see Section 4.2 for details).

$P_j(a, b)$ - denotes a joint distribution between two consecutive requests a, b such that $a, b \in \Omega$.

$\mathcal{L}(s_i|C_j)$ - given probability distribution of a cluster, likelihood of observance transitions in s_i inside cluster C_j is denoted with $\mathcal{L}(s_i|C_j)$.

KL - differences between two distributions are calculated with Kullback-Leibler divergence [41].

JSD - However, KL divergence is not symmetric, nor does it satisfy triangle inequality. As a result, Jensen-Shannon divergence (JSD) is used; see [20] for details.

$\eta_{j,t}$ - is a threashold of **JSD**, under which two clusters are merged together (i.e. when $JSD(P_j||P_t) \leq \eta_{j,t}$), and and $1 \leq j < tleqk$.

$d_M(v_i, \boldsymbol{\mu_j})$ - denotes a Mahalanobis distance between a vector of average categorical requests v_i observed inside sequences of requests s_i, and an average vector of categorical requests $\boldsymbol{\mu_j}$, observed in cluster C_j.

m_i - denotes a vector of average Mahalanobis distances in cluster C_i.

I(a;b) - represents a value of mutual information between two different consecutive requests, such as $a, b \in \Omega$ and $a \neq b$.

i_j - stands for a two dimensional vector of maximum and minimum value of mutual information observed in cluster C_j.

$\Lambda_{\substack{\mathbb{L} \\ \mathbb{A}}}$ - denotes a value of Bayes Factor, in other words, a ratio of evidence supporting legitimate nature of a sequence (i.e $\mathbb{L}$), comparing to the counter evidence (i.e. anomalous nature denoted with $\mathbb{A}$)

$\lambda_{\cdot}$ - $\lambda_{\cdot}$ is a threshold above which a connection is marked as anomalous. In general there are two thresholds depending on range Bayes factors are measured. $\lambda_{\mathbb{S}}$ is used during "soft" classification, while $\lambda_{\mathbb{H}}$ is used during "hard" classification (see below for details).

H_H - denotes a set of clusters for which "hard" detection techniques are applied (i.e. more *legitimate* hosts are classified as *attacking*). These clusters should be frequently visited by attacking hosts, therefore the negative impact on legitimate users is of less importance.

H_S - denotes a set of clusters for which "soft" detection techniques are applied (i.e. more *malicious* hosts are classified as *legitimate*). These clusters are populated by legitimate users, therefore detection thresholds should be relaxed there.

$\ell_{j,r}$ - it is a matrix containing log-likelihood values for each cluster (i.e. $1 \leq j \leq k$) and between $r_\wedge$ and $r_\vee$ requests.

List of Publications

1. Pawel Chwalinski, Roman Belavkin, and Xiaochun Cheng. Detection of Application Layer DDoS Attack with Clustering and Likelihood Analysis. In *Globecom 2013 Workshop - First International Workshop on Security and Privacy in Big Data (GC13 WS - BigSecurity)*, Atlanta, USA, December 2013.

2. Pawel Chwalinski, Roman Belavkin, and Xiaochun Cheng. Detection of Application Layer DDoS Attacks with Clustering and Bayes Factors. In *International Conference on Systems, Man, and Cybernetics (SMC), 2013 IEEE, October 2013*.

3. Pawel Chwalinski, Roman Belavkin, and Xiaochun Cheng. Detection of HTTP-GET Attack with Clustering and Information Theoretic Measurements. In *Proceedings of the 5th international conference on Foundations and Practice of Security*, FPS12, pages 45-61, Berlin, Heidelberg, 2013. Springer-Verlag.

Acknowledgments

First and foremost, I wish to thank my advisor and mentor Dr Roman Belavkin, who has given me great inspiration in virtually everything I have done at Middlesex University. Roman has had a tremendous influence on my perception of intrusion detection and statistical analysis. We have spent together numerous evening, and he has always had time for reading mine yet-another report.

Similarly, I am greatly indebted to Dr Xiaochun Cheng for his creative ideas and contributions. The encouraging criticism at the early stage of this research allowed me to understand the shortcoming of my preliminary work.

Also, I would like to express my gratefulness to the Middlesex Research Office, and especially to Professor Richard Comley, for showing belief in me and offering a studentship. My research would have been impossible without the help I have received from the University.

Moreover, I would like thank you Luis Campo Giralte from University Rey Juan Carlos. Luis was *the only* author working on similar problem who agreed on sharing his data set with me. Therefore, his contribution to this research is greatly appreciated.

I am grateful to all my Friends from TG22 who have shared the moments of happiness and disappointment with me throughout the four years I have spent at Middlesex University.

Moreover, I am grateful to Michal (feksio) who has taught me more about programming over beer and during night chats, than any other teacher.

Lastly, I would like to thank my fiancée Marta for her truly and loving support during time when I could not really be there for us.

Declarations

The candidate confirms that the work submitted is his own and that appropriate credit has been given where reference has been made to the work of others. This copy has been supplied on the understanding that it is copyright material and that no quotation from the thesis may be published without proper acknowledgement.

October, 2013
Pawel Chwalinski

Abstract

Unsolicited web browsing denotes illegitimate accessing or processing web content. The harmful activity varies from extracting e-mail information to downloading entire website for duplication. In addition, computer criminals prevent legitimate users from gaining access to websites by implementing a denial of service attack with high-volume legitimate traffic. These offences are accomplished by preprogrammed machines that avoid rate-dependent intrusion detection systems.

Therefore, it is assumed in this thesis that the *only* difference between a legitimate and malicious web session is in the *intention* rather than physical characteristics or network-layer information.

As a result, the main aim of this research has been to provide a method of malicious intention detection. This has been accomplished by two-fold process. Initially, to discover most recent and popular transitions of lawful users, a clustering method has been introduced based on entropy minimisation. In principle, by following popular transitions among the web objects, the legitimate users are placed in low-entropy clusters, as opposed to the undesired hosts whose transitions are uncommon, and lead to placement in high-entropy clusters. In addition, by comparing distributions of sequences of requests generated by the actual and malicious users across the clusters, it is possible to discover whether or not a website is under attack.

Secondly, a set of statistical measurements have been tested to detect the actual intention of browsing hosts. The intention classification based on Bayes factors and likelihood analysis have provided the best results.

The combined approach has been validated against actual web traces (i.e. datasets), and generated promising results.

Chapter 1

Introduction

There is a handful of different types of attacks designed for almost each layer of OSI network model. In principle, network communication employs seven abstract layers: Physical, Data Link, Network, Transport, Session, Presentation and Application. In this thesis, the focus is put on the application layer and Hypertext Transfer Protocol (HTTP) protocol.

HTTP protocol is used to make information stored on web servers readable for human users. In parallel with great usefulness, the openness of HTTP put Internet users in jeopardy of being victim of an attack generated by hackers. One of a Distributed Denial of Service (DDoS) attack is HTTP-GET attack, also referred to as Application-Layer attack. This attack is based on the transmission of GET requests, sent by computer hosts. In theory, web-browsing is based on GET requests generated by sender. For two-node communication, sender repeats sending GET requests to receiver who replies with requested data. This simplicity provides a way of exhausting physical limitation of a web server. Computer criminals might open a large number of connections in order to reach the maximum number of available slots at the server.

Scenario for running the HTTP-GET attack is similar to other standard techniques used by attackers performing DDoS attack. At first, a number of hosts are infected with a computer virus, and attackers take control over the number of *zombies* (i.e. infected machines). Subsequently, attackers generate a large volume of traffic directed at victim's website with the application of infected hosts, which continue to send GET requests. While the number of compromised systems grows, hackers manage a network of computers called *bot net*. Moreover, this network can be used to provide personal and sensitive information (e.g. bank records) by eavesdropping on one's activity. In fact, bot nets can store up to 70GB of sensitive data

(i.e. user-names and corresponding passwords, bank details etc.) within one week [85]. Besides, popular security company Symantec has recently reported that hiring a bot net of nearly 6,000 machines costs only 350$ per week [34].

Therefore, it is assumed in this thesis that computer viruses will prevail and remain present. It stems from the fact that security policies are either not imposed or ignored. As it has been observed, viruses make up for the underlying mechanism that helps computer attackers pursue their actions. In addition, outburst of computer viruses is reported on a daily basis, leading to the conclusion that a chance of creating a bot net by criminals is reasonably high. Moreover, the flaws in popular operating systems and software application are discovered on a daily basis as well. Thus, infection is an inseparable factor of networked computer systems, and should be stated as a fact. The problem that is tackled in this thesis is not related to virus detection or prevention; neither it is related to malicious code propagation. On the contrary, a potential detection of flooding hosts has been investigated. If a web server is exposed to this attack, it experiences a surge of legitimate GET requests. The difficulty in detection of this behaviour stems from legitimate nature of attacking hosts. Rather than sending ill-formatted network packets, attackers make their zombies comply with computer network regulations, and mimic genuine activity.

Indeed, some researchers claim that the *intention* is the only difference among connections [102]. In addition, because of stealthy appearance of the zombies, their activity becomes indistinguishable from legitimate data flow [80]. Moreover, detection mechanisms based on traffic characteristics become invalid [101]. Yet another obstacle with HTTP-GET attack is a ubiquitous problem in intrusion detection called *flash crowd* effect. It is a situation when many legitimate users visit a website in a short time. As a result, the web-server experiences the influx of new hosts, but it should be recognised as lawful activity.

Moreover, another branch of research closely related to Application-layer attack is detection of unsolicited crawlers. In general, crawler is a reference to an automatic process of extracting information from a website. Numerous business rely on popularity of their website. As a result, bots (i.e. crawlers) from popular web search engines (e.g. Google bots) are more than welcome, as their visits increase popularity in the corresponding search engines (i.e. Google). Moreover, there are link-checkers, that validate web structure saved by web administrators. However, there are multiple instances of unwanted bots. Tan et al. [88] divides malicious crawlers as: (i) shop-bots, (ii) e-mail address collectors and (iii) off-line browsers. The first group is used to automatically process information at shopping website for different purposes. Evidently, these might be betting crawlers (i.e. searching for

best offers of different products) or deal searchers. Similarly, (ii) search for e-mail addresses to build a spam database. Lastly, (iii) are used to download entire website contents. Sometimes, (iii) are referred to as applications for site mirroring. Observe that crawlers do not attempt to exceed or harm servers. On the contrary, by legitimately accessing a website, their harmful act is the *result* of their lawful activity.

Therefore, in order to detect anomalous activity the following are assumed:

1. The focal point of this work states that it is possible to distinguish among legitimate and illegitimate sessions by looking at patterns of browsing behaviour. In other words, during a session, human agents visit different categories in similar pattern.

2. Attackers do not have access to the log of a website, and are unable to repeat valid (i.e. popular) legitimate transitions.

The above conditions imply the following postulate. Essentially, it is assumed that current *interest* (i.e. frequently observed transitions between hyper-links in most recent past) of actual users varies in time, and following (2) from above, it provides basis for distinction of malicious agents. As it transpires in this thesis, actual users visit the same pages, at similar times. As a result, because the interest changes, and the legitimate users generate that process, their transitions will correspond to the actual interest of users browsing a website. Therefore, by concealing web log from attackers , their preprogrammed machines fail to replicate the. This process is described in the following section.

1.1 Access Matrix as Interest Measurement

Based on the assumptions introduced above, the following scenario is presented. Supposed there is a website with 4 hyper-links only, namely: *home.html*, *download.php*, *about.html*, *email.php*. Having analysed a log of clicks generated by actual users, an access matrix has been obtained representing how often a pair of consecutive requests is observed. This is presented in Fig. 1.2. One can note that *home.html*, *about.html* and *email.php* are more frequently visited comparing to *download.php*. However, patterns recorded and presented in Fig. 1.2 are not distinctive enough to provide classification. In other words, randomly selecting bots could easily be perceived as a legitimate user, as the distribution of requests in Fig. 1.2 is almost flat.

Therefore, suppose there is a way to split the access matrix in to a number of clusters. In the scenario above the access matrix is divided into three clusters.

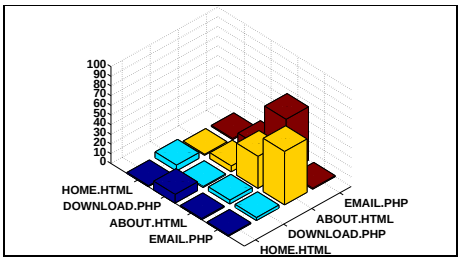

Figure 1.1: Cluster C_1

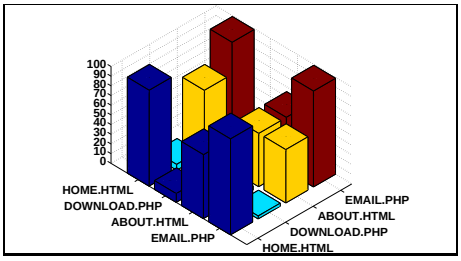

Figure 1.2: Access Matrix

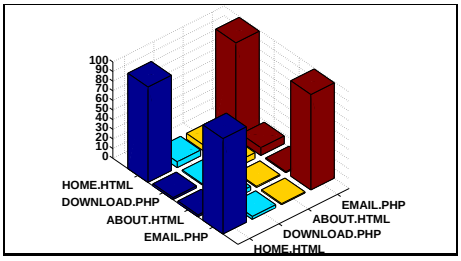

Figure 1.3: Cluster C_2

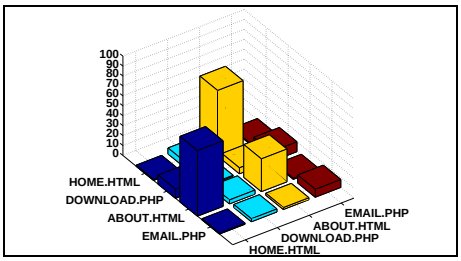

Figure 1.4: Cluster C_3

Figure 1.5: Division of access matrix into a number of clusters.

Clearly, patters of requests observed in Fig. 1.1,1.3,1.4 provide more insight into actual *interest* of legitimate users. Therefore, the main aim of this thesis is to develop an algorithm that forms unique and popular interest groups (i.e. patterns, as the ones presented in Fig. 1.1,1.3,1.4). Subsequently, a measurement should be introduced that on a per-cluster basis, detects the malicious or legitimate intention of a host depending on information encoded in the set of clusters.

These aims formulate research question presented in the following subsection.

1.2 Research Question

Having introduced the attack and its intention dependency, this research tries to address the following question: *Can unsolicited web browsing be detected without temporal information by means of clustering and statistical measurements ?*

In general, the above question can be broken into the following questions:

1. Is there a clustering algorithm that allows for discovery of recent patterns of interest of legitimate users?
   (a) Is there a metric that explains the relation among visiting pages?
   (b) Is the clustering good enough for attackers not to discover it?
   (c) Can clustering provide insight whether there is a malicious attempt taking place at the server?
2. What kind of statistical test can be used to detect whether or not one connection is attacking or legitimate, regardless of strategies chosen by attackers?
3. Can an attack be detected in real time?

These questions have been studied in this thesis, and the answers are provided as the contribution to knowledge. To provide answers to the formulated questions, the large task has been divided into following stages:

1. **Data Collection (Acquisition)** - For the purpose of this research four data sets have been obtained (described in detail in the following chapters). Specifically, given different ways that web data sets have been recorded, web interest is broken into two types: (i) macro-interest and (ii) micro-interest approaches. While (i) analyses interest among web categories, (ii) is a more detailed approach, and considers relations among hyper-links.
2. **Data Preprocessing** - It is observed that current web pages contain numerous web objects, each identified by its Uniform Resource Identifier. As a result, before analysis of (ii) is performed, a method of dimensionality reduction is introduced, based on popularity of object requests.
3. **Feature Selection** - Given the assumptions about the regular transitions of legitimate users, the main feature to consider will be a transition between object of interest during session. As a result, it has been shown that a *change* in the expected order or number of transitions is indicative enough for classification.

4. **Clustering** - A clustering algorithm is introduced to group users sequences of HTML requests, which express similar interest. The algorithm allows for discovery of recent pattern of behaviour that remain unknown to attackers. As a result, by failing to adapt to the clustering scheme, attacking hosts indicate their presence by making unlikely transitions inside a cluster.

5. **Anomaly Measurement / Classification** - Once the clustering scheme has been violated, and the attacking attempt discovered at a server, a set of statistical measurements is introduced to measure whether one sequence of requests is malicious or lawful. Ideally, it is expected to have one measurement that should suffice for intention detection, regardless of different browsing strategies, implemented by attackers. Moreover, this measure should be scale-independent (i.e. undesired browsing by a single crawler should be notified as well).

6. **Results Analysis** - The *only* techniques for improving performance of classification are either by extension of feature space or improved data set decomposition, or both [46]. As a result, discussion of both is provided to picture future ways of research to improve this work.

1.3 Contribution to Knowledge

In general, the detection of unsolicited web browsing comes down to intention classification, and in fact it is the main problem of this research. In general, there are two main approaches based on authentication [80] and profile based approaches [102, 82]. The authentication methods attempt to recognise malicious user by protocol modification or presenting various challenges for arriving connections [4]. On the contrary, the other methods focus on recent patterns of activity and compare incoming connections against the profile.

This research falls into the latter category, and a clustering approach is introduced to group sequences of requests representing similar behaviour. The algorithm is motivated by the work of Barbará [3], where entropy minimisation algorithm has been used for categorical data streams. However, in the original form, the algorithm does not allow for clustering of sequences, which are built on a sample space with numerous events (e.g. visiting a link on a e-commerce website), due to computational constraints. Therefore, the modified version of the algorithm is the main contribution that allows to detect whether or not the arriving traffic contains any attacking sequences.

In addition, a measurement based on likelihood analysis is introduced. This measure is: attack-strategy independent (i.e. is able to detect different browsing strategies chosen by attackers), and attack-scale independent (i.e. regardless of the number of browsing bots, the attempt is discovered, especially when this number is very small).

1.4 Organisation of the Thesis

The above mentioned considerations have been organised in the following structure

- **Chapter Two - Theoretical Background** - an introduction to intrusion detection field is given in this chapter. Moreover, the exact scenario of running an Application-Layer attack is provided.
- **Chapter Three - Literature Review** - in this chapter, the gaps of current detection techniques identified, and the research question is formulated.
- **Chapter Four - Methodology** - the mathematical apparatus applied in the clustering mechanism and statistical measurements are described in the methodology chapter. Mathematical formulation and implementation of the algorithm is described here. Moreover, statistical measurements have been introduced to measure intention of arriving hosts.
- **Chapter Five - Results Analysis** - in this chapter, analysis of the clustering algorithm and detection performance of statistical measurements are provided.
- **Chapter Six -Discussion and Future Work** - advantages and disadvantages of the approach, and comparison against existing solutions are given in this chapter. Moreover, this thesis concludes with analysis of work that should be done to improve this research.

Chapter 2

Theoretical Background

An Intrusion Detection System (IDS) is a tool that attempts to discover intrusion that it is taking place in a system. Suppose that IDS analyses current activity of users in an arbitrary system. Users' actions can be thought of as packets sent from one host to another, web pages visited during a web session, Structured Query Language (SQL) queries sent to a database server, and others. Note that each entity belongs to a set with unique elements. For example, entity web page belongs to a set with domain relatively different than a set containing possible SQL queries. As a result, IDS monitors sequences of entities generated from each set, and searches for suspicious activity. Moreover, users' actions are represented as sequences of actions, and are grouped in a data set D of current activity in a system. Each variable-length tuple $s_i, 1 \leq i \leq |D|$ inside D, denotes a sequence of i-th agent, such that $s_i = (s_i^1, s_i^2, \ldots, s_i^j, \ldots, s_i^{n_i})$, where n_i denotes a number of actions that i-th user took. Each s_i^j should be thought of as *information* related to j-th action inside sequence s_i (e.g. j-th request of web page, j-th packet in a network stream, etc). It is assumed that information encoded in sequences of requests is sufficient for classification.

Note that the intention of activity could be either peaceful or harmful. As a result, for the binary intention, IDS employs a classification function $f : D \to \{-1, +1\}$ that assigns a label $y = +1$ to a potentially legitimate sequence, and $y = -1$ is assigned to malicious sequences. N represents the maximum length of sequences inside D. Thus, given input data s_i, IDS tries and assigns the class $y_i \in \{-1, +1\}$. In addition, given the proportion of pairs $\{(s_1, y_1), (s_2, y_2), \ldots, (s_{|D|}, y_{|D|})\}$ assigned to one class or the other, it is possible to evaluate whether a system is experiencing the attack, or whether the current users are legitimate.

Observe that the motivation behind using IDS is existence of legitimate pat-

terns inside D, and their *regularity*. Traditionally, it is assumed that Denning [13] introduced the concept of IDS, which was based on searching for deviations from a *regular* pattern of legitimate users. On he other hand, attacking hosts differ in behaviour. In general, IDS searches for hosts that send multiple request, resource-intensive SQL queries, or even attempt to download heaviest images from a website. These actions, are rarely observed in D, and are blindly assumed as deviations. Surprisingly, the ideas formulated by Denning in 1987, have not changed at all, and current IDS's follow the concepts introduced in [13]. However, as it transpires in this chapter, it is very challenging to define what malicious really is, in the area of unsolicited web browsing.

In the following section, a detailed description of the attack is provided. Initially, it is shown how the attack was brought to life. Specifically, different types of attack and its targets are presented; followed by typical victims, and deployment of their IDS's. Moreover, difficulties related to the attack detection are demonstrated. In principle, a sample of code is provided from the actual bot net to show how easily the attack can be triggered. This concludes theoretical description of the attack.

Subsequently, the focus is put on research in IDS domain. Initially, a comparison is drawn among other approaches against unsolicited web browsing. The chapter concludes with identification of existing gaps in the contemporary field, and motivation for the current research.

2.1 Different Types of Denial of Service Attack

It is well known that network communication can be divided into 7 abstract layers with ISO/OSI model [90]. These are: Physical, Data link, Network, Transport, Session, Presentation, and Application layer. In general terms, data created (with some computer software, i.e. *application*) by user in Application layer is subjected to different form of formatting and encapsulation, before it is received (in the corresponding application) by receiver. As it has been mentioned in Chapter 1, HTTP protocol is located in Application layer. However, historically, DDoS attacks originated from the lower layers, because popular application (e.g. web browsers) were introduced later in time. One of the very popular DDoS attacks, SYN Flood (explicitly described in RFC 4987[1][17]), is based on large volume of TCP SYN packets sent to receiver, and taking place in Transport layer. SYN Flood was formally introduced in 1994 [17], and research community started working on counter measurements in 1997, when pioneered by Schuba et al. [75]. In general, the attack misuses a re-

[1] `http://www.ietf.org/rfc/rfc4987.txt`

quired step in TCP/IP transmission establishment. The establishment process for TCP/IP and is referred to as TCP three-way handshake [90].

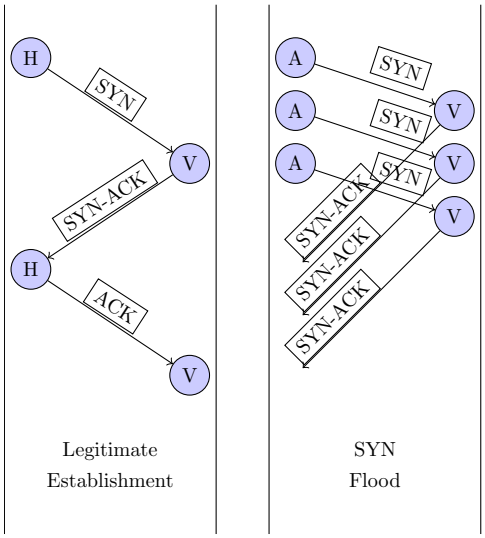

Figure 2.1: Internet attackers have learned to manually modify packets, to run SYN Flood attack. As a result, they reserve resources with non-existent IP addresses exhausting the server's resources.

A human agent (denoted with H in Fig. 2.1) sends a synchronising packet (i.e. SYN), that is confirmed by the victim (denoted with V in Fig. 2.1) (with a SYN-ACK), and the agent starts sending data with a subsequent confirmation (with ACK) of the first victim's packet. This scenario is presented on the left-hand side of Fig. 2.1. The Internet attackers (denoted with A in 2.1) can violate this mechanism by opening multiple slots at the victim site, and never confirming them, what has been presented on the right-hand side of Fig. 2.1. Note that if the victim knows the network address of attackers, then the addresses might be taken note of, and stored as malicious. However, attackers might randomly generate arbitrary addresses, and send packets to the victim. This way, the victim keeps the resources reserved for a non-existent address. In addition, this might lead to exhaustion of resources, and success of the attackers.

The way that the DDoS attack against the HTTP protocol is run, is alike with the scenario described above. To understand attackers' behaviour a copy of Agobot virus [63] has been analysed. It is one of the most famous virus instances that has been studied by the research community. The most interesting features of the code has been presented in Listing 2.1. Note that the code portion is just a small excerpt from numerous functions, and for presentation only.

Listing 2.1: Portion of Agobot's code

```
1  /* ..... */
2  REGCMD(m_cmdUDP,"ddos.udpflood","starts a UDP flood",false,this);
3  REGCMD(m_cmdSyn,"ddos.synflood","starts an SYN flood",false,this);
4  REGCMD(m_cmdHTTP,"ddos.httpflood","starts a HTTP flood",false,this);
5  REGCMD(m_cmdStop,"ddos.stop","stops all floods",false,this);
6  /* ..... */
7  g_pMainCtrl->m_cIRC.SendFormat(m_bSilent, m_bNotice, m_sReplyTo.Str(),
8  "%s: flooding %s port %u, %u times, %d ms delay.", m_sDDOSName.CStr(),
9  uURL.sHost.CStr(), uURL.iPort, m_iNumber, m_iDelay);
10 /* ..... */
11 char *g_szUserAgents[]={
12 "Googlebot/2.1 (+http://www.googlebot.com/bot.html)",
13 "Mozilla/4.0 (compatible; MSIE 4.01; Windows 95)",
14 /* ..... */
15 int iUserAgent=brandom(0, iNumUserAgents);
16 /* ..... */
17 DoHTTPRequest(sSendBuf.CStr(), uURLTemp);
18 /* ..... */
19 int iSleep; if(!m_iDelay) iSleep=brandom(3600000, 86400000); else
       iSleep=m_iDelay;
20 /* ..... */
```

Once complied, the bot master is allowed to pick what type of attacks to perform (lines 2-5). Subsequently, a command is sent over a hidden chat channel indicating which host should be attacked (lines 7-9). Note that parameters like the number of requests or delay can be specified (line 8). Subsequently, a bot picks up the parameters of the attack and begins initial configuration. Initially, a bot will try to mask its presence and choose a random (line 15) agent configuration (e.g. as a Googlebot, or a popular web browser; see lines 12-13). As a result, the victim host will believe that transmission is carried out with a valid host. Subsequently, the bot can request different objects (images, scripts, HTTP pages, etc.) in a loop. Once the number of requests has been exhausted (specified in line 8) the bot remains idle. This inactivity time can be set up manually (line 9), or is predefined as a time period between 1 (3600000 milliseconds) and 24 (86400000 milliseconds) hours (line 19).

The design of the code implies the following issues. First of all, the attack is run by sending HTTP request. As a result, transmission is always established, and confirmed by both parties. Therefore, the victim knows the network location of a bot. Secondly, attackers can specify physical parameters of actual connection. As a result, it is trivial to run the attack at a lower rate, so that bots are not detected

by a rate-dependent IDS. In addition, the bots are programmed to act as legitimate users. Therefore, there are no deviations in semantic network regulations. Lastly, the most appalling argument is the inactivity period set by the attackers. If not chosen manually, it is being drawn from 1 to 24 hours. This implies that attackers assume to have a large bot net and each of their zombies "can afford" to remain inactive for up to 24 hours.

It is difficult to estimate the number of bots that a bot net consists of, as discussed by Rajab et al. [69]. One of the reasons is the fact that infected hosts can be members of different bot nets, as observed by Zhuge et al. [120]. In addition, bot masters keep on changing their command and control (C&C) channels [85]. Indeed, one of the possible medium for a C&C channel is application of Internet Relay Chat (IRC) protocol. Observe, that bots using Agobot application will communicate with the bot master using an IRC channel (see line 7 in Listing 2.1). Nevertheless, depending on activity of actual hosts (i.e. being either on or off), the number of hosts participating in attack can vary from few thousands to several hundred thousands [85]. Therefore, suppose that bots have been programmed to visit a page every ten minutes on a day or around 144 times per day. Assuming a 500,000 strong bot net, a victim website might experience around 833 connections constantly browsing its content. Obviously, these assumptions are very benign, and as Gu et al. [29] have pointed out, traffic generated by bot nets can reach up to few Gigabytes in few minutes.

Having analysed actions taken by attackers, the following sections explain the attack, once the bot net has been activated. In principle, different approaches to running the attack are presented. Moreover, different deployment locations of IDS are considered.

2.2 Scenario of Denial of Service Attack

Paxson [65] observed that there are two main approaches to the attacks, defined by the types of nodes that attackers might use. The first one is the previously introduced DDoS attack, and depicted in Fig. 2.2. The other one is Reflector DDoS (RDDoS), and presented in Fig. 2.3. In order to judge whether or not the attack is taking place, IDS is placed in different physical location of a networked environment. In the subsequent session, a network design is provided with description of where detection can take place. In addition, different DDoS targets (i.e. victims) are described in detail.

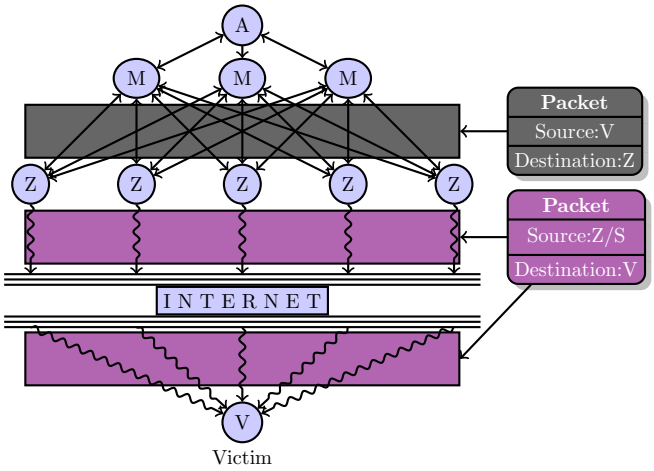

Figure 2.2: Simple scenario of DDoS attack.

From Figure 2.2 one can understand the difficulty of DDoS attack detection. Initially, attackers infect and take control over a number of computer hosts. Subsequently, a smaller portion of the nodes is transformed into master nodes (denoted with M). These are used by attackers to remain anonymous, and refrain from directly communicating with zombies (denoted with Z). On the other hand, the remaining nodes are used for performing the attack, and are transformed into zombies. Because of the way TCP/IP protocol is designed, attackers are able to manually fill up arbitrary fields of packets. Therefore, master slaves transmit many packets to zombies with source field set up to the victim address, and destination set the a particular zombie's address. This process directs following replies at the victim. In principle, according to the protocol regulations, zombies reply to the victim to revoke communication. Moreover, zombies might be programmed to *spoof* (i.e. randomly generate) their physical address (denoted with capital S in the source field). By doing so, victim replies to likely non-existing destination, and awaits for incoming packets. In addition, the resources are being reserved at the victim's site for some limited time in case the transmission is valid, and with an actual node. Note that if address is not spoofed, then victim is able to locate attacking hosts. However, if attackers manage an enormous bot net, and anonymity of their hosts is not required, they prefer to establish communication with the victim.

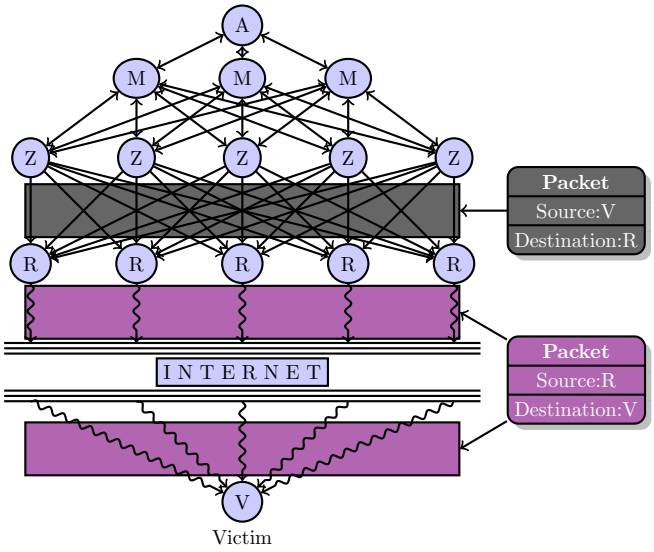

Figure 2.3: The application of Reflectors introduces challenging level of detection difficulty.

Another form of the attack incorporates an additional layer of zombies called reflectors, and is presented in Figure 2.3. In this scenario, attackers are happy to reveal the identity of reflectors. As it has been noted reflector can be any host that returns a packet, if sent a packet [65]. Therefore, the attack might utilise any server, and any machine that can provide two-way communication.

As for now, it becomes obvious that detection of RDDoS is extremely challenging. Clearly, it can be based on the approach that requests sent by zombies are regular enough to implement semantic classification, which assumes traffic regularity [65]. In addition, detection of spoofed addresses can be resource-intensive. Ferguson et. al. [22] observed that by monitoring traffic in centralised locations it should be possible to filter out malicious traffic. Clearly, if a router observes a randomly generated address field that cannot be coming from its address pool, then the packet should be dropped. This form of attack prevention is called *egress filtering*[16]. Yet another method for detection of spoofed addresses might follow simple logic, and the following condition. Suppose that every incoming packet comes from a spoofed-address host. Therefore, any resource reservation takes place once a connection has been established (i.e. a receiver knows that a sender is a valid host, and they have exchanged packets with confirmation of their authenticity). This method has been developed by Brenstein et. al. [12], it is called SYN-COOKIE, and its implementation has been introduced into FreeBSD Unix [48].

Note that detection can take place at different points of network topology.

Clearly, from a source of attacks to the victim's destination there are multiple locations at which detection can be performed. These are described in the following section

2.3 Location of Detection Systems

According to Mirkovic at. al. [56] defence mechanism can be placed in three main locations: victim network, intermediate network and source network. Figure 2.4 shows the scenario of RDDoS attack and possible deployment locations. Chang et. al. [8] breaks intermediate network into: upstream ISP network and victim's ISP network.

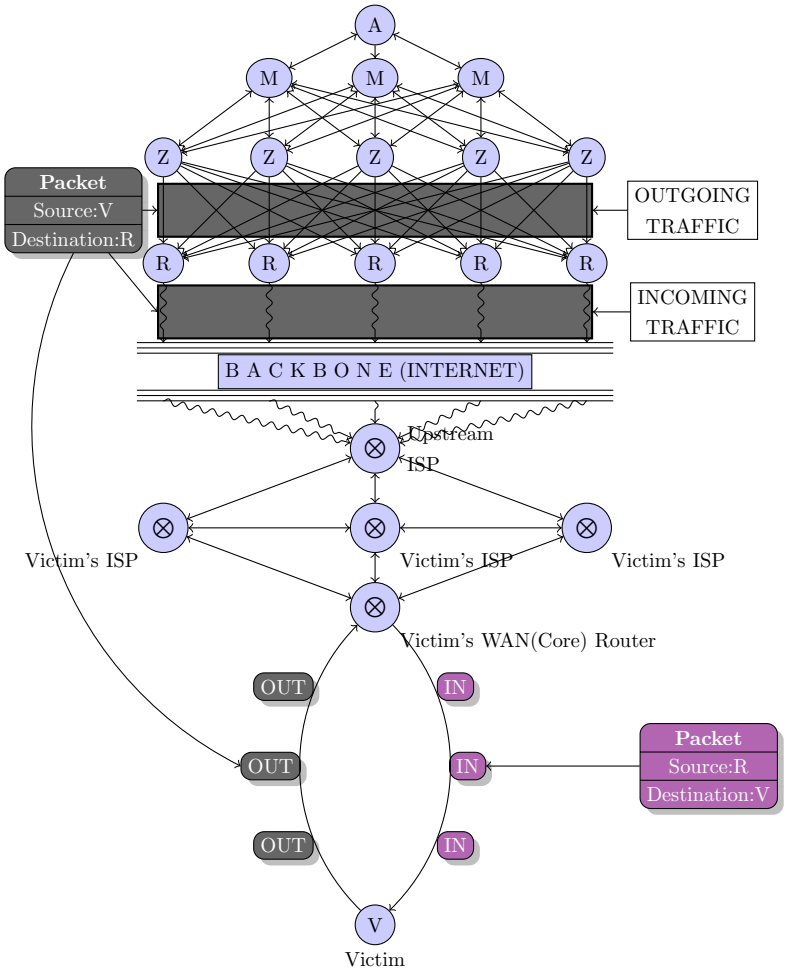

Figure 2.4: Packet traversal from attacking hosts to victim.

Clearly, because victim network is always the ultimate target of the attack,

it should be relatively easy to discover malicious activity. This deployment should be placed at victim's Wide Area Network (WAN) router in Figure 2.4. Below the router there are two streams of packets denoting incoming (denoted with IN) and outgoing (denoted with OUT) packets. Note that the addressing scheme is the same for: (i) *outgoing* traffic from Z nodes to R nodes; for (ii) *incoming* traffic from R nodes to victim V; (iii) and the *outgoing* traffic from victim V to reflectors. It stems from the fact that zombies trigger transmission by sending a *response* to reflectors, which reply to the victim with actual packets. The arriving packets are correctly marked from reflectors to the victim (see purple packets in Figure 2.4). The victim replies, and outgoing packets are denoted with the victim's source address and destination of a reflector's address. The victim's WAN router let packets through because of their legitimate nature (i.e. the packets follow network regulations), and because the source address corresponds to the memory-stored address of the victim. Therefore, to some extent one should argue that the easiest (yet relatively difficult) detection process should take place at the victim's core router.

As it has been described in the previous section and referred to as egress filtering, detection could take place close to the source network. This is just before the backbone network denoted with blue rectangle in Figure 2.4. Again, egress filtering should readily filter out the packets with spoofed source addresses. However, reflectors are physical machines with valid addresses, and therefore this method fails. The only solution is to assume there are some similarities in attacking traffic of reflectors before it enters the backbone network.

The main problem with filtering traffic among routers inside backbone networks stems from speed and amount of data. Current Internet traffic, at Gigabyte level with million of computers around the world, makes filtering implausible [8, 56]. For example, one can come up with opposite of the egress filtering, and block all packets whose source addresses have never been seen at routers, or are not allowed to travel between two nodes. This process is referred to as *ingress filtering*[22]. Because it protects backbone autonomous systems, it has to be planted in the backbone network. However, the computational cost related to detection at this level, makes it difficult for implementation; moreover, it slows down router performance significantly as observed by Geng et al. [26]. Therefore, these solutions yet effective are impractical because of the overhead in computations. Therefore, most of the learning systems are placed very close to the victim's site. However, technical solutions presented in Section 3.1.4 vary from local to global solutions.

2.4 Victim Types of DDoS Attack

There are 5 main targets of network criminals [56]: 1.Host, 2.Resource, 3.Network, 4.Infrastructure, 5.Application. The methods and specifications of each attack have been provided below:

1. *Host attacks* are targeted at a single host in order to reboot it or freeze it for some time. There are many reasons behind attacking a single host. Mainly, the attacks targeted at one host are part of a larger-scale operation. A host could be a Linux-based server with multiple users that access multiple applications in a shell environment (i.e. by typing in commands). Yeung et al. [108] developed a method of users discrimination based on statistical analysis of the sequence of commands and their likelihood of occurrence. Similarly to Yeung, Ye et al. [107] have performed statistical analyses of system events. Therefore, in general, detection for host based system comes down to two main sections: building profiles of legitimate behaviour from past data (e.g. system log), and feature selection that allow for user discrimination.

2. *Resource attacks* are targeted at crucial resources of a fretwork, for example by halting a database server, access to a website will be limited. The solutions vary from resource replication [56], through implementation of intrusion-tolerant protocols as developed by Cachin at al. [7], to honey pots against DDoS introduced by Weiler [96] (i.e. traps for scanning or attacking hosts).

3. *Network attacks* are against a portion of network subnet. Historically, it was possible to detect these attacks at edge routers (see WAN router in Figure 2.4) by using application support. One of the first solutions of these types was Snort, developed by Roesch [73] that is still one of the most famous IDS. Subsequently, because of the amount of network traffic, the solutions have moved from a centralised point to a more distributed approach as explained by Vigna et al. [92]. However, in reality, detection and prevention mechanisms are deployed at the backbone level as noted by the authors of[8, 56], and victims are required to seek help from ISP.

4. *Infrastructure attacks* are directed at the key components of the backbone network - the Internet. These involve attacks against global routers or domain name servers (DNS), as reported by Kambourakis et al [37]. Essentially, switching off DNS servers enforces using physical addressing (e.g. `http://123.123.123.123`) instead of a "name" encoded in DNS server (e.g `www.google.co.uk`). However, the counter measurements involve cooperation of

backbone ISP administrators [56]. On the other hand, there are methods to develop a new architecture against Infrastructure DDoS attack, as initially proposed by Wan et al. [93].

5. *Application attacks* are attacks against one particular application running on a system. As a result, by disabling application attackers prevent legitimate users from accessing its resources (e.g. website content). As it has been described in Section 1, and reported in [56] application attacks are the most difficult to detect because of the legitimate nature of the attack. In principle, a victim observes incoming legitimate traffic that follows network regulations. Therefore, the only way to stop the attack is by discovering patterns of legitimate users, and hoping that attacking hosts are unable to repeat legitimate behaviour. Historically, the attacks were deployed against ICMP protocol[56], a diagnostic protocol used to communicate on errors existing in networks. However, Moore et al. [57] have observed that attacks against ICMP make up for only 2% of all DDoS conduct observed in the wild. The other Application attack is HTTP-GET attack, that has been analysed in this thesis. Moreover, El Defrawy et al. [19] have shown that there is another attack that is using different types of reflectors. In their work, they have presented how to direct Bit Torrent (i.e. popular P2P application for sharing purposes) traffic at arbitrary host.

Clearly, victims described above detect attacks by noticing increased resource consumption. Moreover, the research of this thesis is focused on attacks against applications. The main motivation behind research in the application layer attacks stems from the legitimate behaviour of attacking hosts, and difficulty of separation among legitimate and attacking hosts. However, when the increased number of legitimate users arrive at a website, their traffic should differ from attacking hosts trying to overload resources. This phenomenon is described in detail in the subsequent section.

2.5 Application Layer Attacks and Flash Crowds

Term *flash crowd* has been introduced by Niven in 1971 [61]. Originally, it described an event in which a large number of people were transformed into places, where interesting scenes were happening. Transformations were possible by transfer booths. However, criminals were able to move as well. Therefore, an interesting event might have attracted a large number of legitimate users, which could have further coin-

cided with the arrival of villains.

Similar behaviour takes place at websites. In particular, president elections [35] or important sport events [101] (e.g. football World Cup) lure multiple users into visiting a website. Subsequently, the web servers experience the increased number of hosts, and bandwidth consumption. Nevertheless, the traffic is legitimate, and should be dealt accordingly.

Comparison of flash crowd and DDoS attack has been provided by Jung et al. [35]. In [35], the authors have tried and found differences and similarities in traffic features. Their main conclusions are formulated as follows:

1. **Stable traffic of legitimate users during flash crowd** - one of the most useful assumptions in research against attacking bots is the fact that legitimate traffic displays similar characteristic before, during, and after flash crowd. As a result, the increased traffic is caused by the increased number of users. Therefore, the increased rate of requests *is not* common during the flash crowd event, on a per-user basis. Clearly, the server notices increase in the net request per second average. However, it is due to the number of users, rather than number of clicks. Moreover, legitimate users *adapt* to a more unresponsive server, and decrease their rate of requests. This has been confirmed by Chen at al. [10].

2. **Flash crowd is caused by known users** - Specifically, Jung has reported that the number of previously seen subnet ID's varies from 42.7% to 82.7%. On the contrary, during DDoS attack (that attempts to mimic flash event) this number corresponds to 0.6% - 14%. This implies that the attack is caused by relatively small number of previously seen users. Moreover, during a flash event the number of Autonomous Systems (i.e. unique IP addresses representing a subnet of hosts) can involve hundreds of AS's, as reported by Wendell et al.[97].

3. **File popularity follows legitimate distribution** - Pareto distribution has been used as a standard to provide distribution of interests varying from most popular to least popular web objects [6]. It is called Zipf distribution when talking about discreet events. Jung has shown that during the flash event, requested files follow Zipf-like distribution. In most general case, as pointed out by Adamic et al. [1], a probability of requesting an i-th documents from a N-document website is proportional to: $P(i) \approx \frac{1}{i^{\alpha}}$ where $\alpha \in [0; 1]$. On the contrary, the attacking hosts request distribution deviates from the expected distribution. Moreover, the attacking hosts request only a small number of

web objects [35]. This assumption of the attacking behaviour is crucial to a definition of strategies chosen in this thesis. Specifically, it will transpire that this strategy will generate the best results, from the attackers' point of view.

These conclusions are cornerstone to most of the previous works described in Section 3.1. Especially the first assumptions allows for distinction among user intentions, as the attacking hosts are assumed to send requests at increased rate.

Moreover, having assumed that known users contribute to flash event, it is reasonable to focus on new locations. However, the current architecture of the Internet implies that addresses behind one IP address will be marked anomalous as well. Clearly, address translation was employed by the overly increasing number of hosts assigned to a network. As a result, a Network Address Translator[2] has been introduced, and formulated by Egevang [18]. In consequence, hosts behind one IP address will be marked with the same address, and traffic separation is dealt with by routers. Therefore, given conclusion (2) from above, it seems reasonable that attackers infect random hosts, whose subnet address has not been seen before. As a result, blocking new subnet might unwillingly block a legitimate user that happens to be a neighbouring host to the attacking host(s).

In addition, given the exemplary zombie's code (see Section 2.1), attackers can possibly implement their bots to comply with (1). Furthermore, before running the attack, they can direct their zombies at a website, to inject attacking subnets into a set of former visitors. However, they will be unable to repeat (3), because it is assumed that attackers do not know the actual interest of legitimate users.
Having introduced the attack, its location and deployment of IDS, and difficulty of flash crowd detection, it is essential to turn attention to IDS and their performance. Predominantly, standard measures of IDS performance are presented that are used extensively across literature.

2.6 Performance of Intrusion Detection Systems

One of the ways to measure IDS's performance is to analyse how often it misclassifies or correctly recognises agents' behaviour. Given that intention of each connection is binary, the system can make a mistake or label it correctly. Note that intrusion detection has been perceived as a branch of pattern recognition field. As a result, traditionally, IDS's performance can be described with four definitions [54] that correspond to four possible states in Table 2.1, often referred to as a two-by-two

[2] `http://www.ietf.org/rfc/rfc1631.txt`

confusion matrix. Observe that according to its definition, IDS searches for *intrusions* (rather than lawful behaviour). Therefore a *hit* (i.e. a success of any IDS) is defined as follows:

Definition 1. True Positive - is an event when actual illegitimate activity is correctly recognised (i.e. when s_i is *malicious* and $f(s_i) = -1$).

The opposite to the above denotes misclassification (of the *positive*), and is explained in the following way:

Definition 2. False Negative - denotes an event when malicious activity is marked as legitimate attempt (i.e. when s_i is *malicious* and $f(s_i) = +1$).

Similarly, for recognition of authentic behaviour, there are two corresponding terms:

Definition 3. True Negative - is an event when legitimate activity is marked correctly as harmless attempt (i.e. when s_i is legitimate and $f(s_i) = 1$).

Conversely, misclassification of rightful sequence s_i is defined as:

Definition 4. False Positive - false positive alarm describes misclassification of legitimate activity as malicious (i.e. when s_i is legitimate and $f(s_i) = -1$).

Subsequently, rates of the above metrics are defined in the following way [21]:

1. **True Positive rate**, sometimes called *hit rate* or *recall* of IDS is estimated with the formula

$$\text{TP rate} \equiv \frac{\#\text{ of positives correctly recognised}}{|\text{positives}|}$$

   where $|\cdot|$ denotes cardinality.

2. **False Positive rate** is often referred to as false alarm rate. Clearly, in IDS domain a false alarm denotes a misclassified *non-positive*, (i.e. see Definition 4)

$$\text{FP rate} \equiv \frac{\#\text{ of negatives incorrectly classified}}{|\text{negatives}|}$$

Other terms that often appear in IDS field are [21]:

Definition 5. Specificity denotes proportion of correctly classified negatives in a test, and calculated as:

$$\begin{aligned}\text{Specificity} &= \frac{\text{True negatives}}{\text{False positives} + \text{True negatives}} \\ &= 1 - \text{FP rate}\end{aligned}$$

Actual \ Predicted	Normal	Intrusion
Normal	True Negative	False Positive
Intrusion	False Negative	True Positive

Table 2.1: Possible outcomes of IDS's decision

Definition 6. Sensitivity is a synonym for recall, i.e. True Positive rate, and defined as

$$\begin{aligned}\text{Sensitivity} &= \frac{\text{True positives}}{\text{False negatives} + \text{True positives}} \\ &= 1 - \text{TN rate}\end{aligned}$$

Definition 7. Precision is defined as positive predictive value, and calculated as:

$$\begin{aligned}\text{Precision} &= \frac{\text{True positives}}{\text{False positives} + \text{True positives}} \\ &= 1 - \text{FN rate}\end{aligned}$$

In general, the most important trade-off for IDS is the relation between FP and TP, and presented in Figure 2.5. A point $(0; 0)$ denotes very "poor" classification; the IDS never makes a wrong guess on legitimate activity, however, it misclassifies all illegitimate agents (TP being equal to 0). This situation might correspond to a system with unlimited resources, and able to provide intrusion-reliant solution. On the other hand, a point $(1; 1)$ represents a very "strict' system for which each sequence is illegitimate, regardless of the actual intention (TP, FP being equal to 1). Clearly, this might correspond to a scenario when the system has been shut down, and awaits to be up again.

Ideally, one seeks the systems for which TP is close to 100% (i.e. when each malicious attempt is recognised), and FP is close to 0% (i.e. no legitimate session is misclassified). This scenario is very difficult to reach, and there are always some miss-hits present in IDS classification. Nevertheless, the optimum performance of any IDS is reached at point $(0; 1)$, when IDS has learned many informative features that allow for perfect classification.

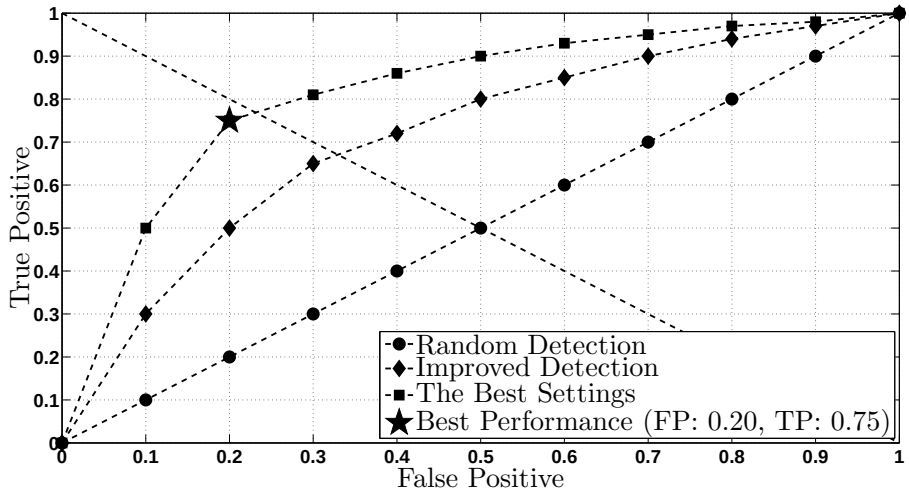

Figure 2.5: Receiver Operating Characteristic curve for three IDS's systems, performing at different

In addition, the performance of IDS can be represented with Receiver Operating Characteristic (ROC) curve that provides insight into overall classifier's accuracy. Bradley [5] and Fawcett [21] have written exceptional tutorials on application of ROC curves in the evaluation of machine learning algorithms.

Suppose that there is an arbitrary IDS, and its performance for three different settings have been presented in Fig. 2.5. Observe, that in this scenario the performance is presented as a *line* (rather than a single point), and it is normally related with performance conditioned on some parameters θ. This can represent a varying number of actions stored in a sequence s_i. Therefore, suppose that $\theta \in \{0, N\}$, where N again denotes the longest possible sequence, and suppose further that $N = 10$. Initially, performance of neither of the systems presented in Fig. 2.5 can classify incoming connections *before* they make a first request (i.e. point $(0, 0)$). For the system presented in Fig. 2.5 short legitimate sequences are popular. This corresponds to a high TP, and relatively low FP. However, the longer the sequences, the more legitimate sequences are misclassified. Furthermore, this comes with a trade-off of a higher rate of intrusion detection.

Moreover, suppose that there is extra information, say time-zone, and is divided into three zones; thus three performance curves in Fig. 2.5. Therefore, rates depicted in Figure 2.5 represent rates conditioned on the parameter $\theta = \{N, t_{zone}\}$.

The first system has been designed to detect intrusions randomly. Its expected detection has been presented with a line marked across with $\bullet$. By looking at the trend of this line, one can learn that on average $TP = FP$. For example, for

nine-request long sequences the system should be able to recognise on average 90% of illegitimate sequences, with the same error rate of legitimate classification. Subsequently, suppose there is an improved IDS and its performance has been denoted with ♦. For this system, the recognition has been slightly improved. Finally, the best system for this particular scenario has been denoted with the best settings line (i.e. the line marked across with ■).

Depending on the attack volume, the varying expected behaviour of IDS is looked for. Essentially, while the number of attacking hosts is relatively large minimising the number of FP is crucial. On the other hand, maximising TP, should be sought if the system is intrusion-tolerant. However, in general, finding the optimum values of TP and FP at the same time is required.

This corresponds to the point denoted with a star in Figure 2.5. The easiest way to find such a point is to connect two points $(0; 1)$ and $(1; 0)$ with a straight line (a dashed line without markers in Figure 2.5). The first intersection with this line (or a point closest to the intersection) will be the best performance of IDS. Because the intersection point with the best setting line is not actual performance point, the point with $FP = 0.20$, and $TP = 0.75$ is chosen as the best performance of the IDS system. Therefore, in order to maximise performance of the IDS, one picks the best performing IDS, and sets its parameters so that detection is performed after second request and the particular time zone. This corresponds to the point denoted with the star.

In the subsequent session, a description is provided of two popular branches of intrusion detection research: misuse and anomaly techniques. Having introduced both concepts, research survey is provided on current techniques tackling unsolicited browsing problem.

2.7 Different Approaches to Intrusion Detection Systems

The most general division of IDS is into two approaches: anomaly and misuse based detection systems [36]. Misuse approaches are often referred to as signature-based solutions because of their dependency on the patterns of known attacks. In general, a system is secured as long as its attack database is updated with the latest, so-called, 0-day attacks. As a result, they are often referred to as signature-based approaches in which each s_i is compared against a database containing suspicious activity. Even though updating the database is a time-consuming task, these systems generate very low false positive (i.e normal $\rightarrow$ normal) ratio. It stems from the fact that harmful

activity is describe with very precise definitions of illegitimate behaviour, as reported by Ghosh et al. [27]. However, even though the database is being instantly updated, the polymorphic or modified versions of known attacks will not be picked up by the detector. Thus, these solutions suffer from detection of novel attacks, and should generate extremely high rate of false negatives (i.e intrusion $\rightarrow$ normal) [27].

On the other hand, the anomaly approaches attempt to initially learn a profile of *legitimate* behaviour. Observe that this is to the contrary with misuse approach where *attacker* strategies are modelled. In addition, there is a strong assumption of the legitimate profile of behaviour being inaccessible for attackers. In other words, consider an IDS for HTTP-GET attack. It is assumed that attackers do not know the patterns of transitions among web pages, or frequently sent SQL queries. Subsequently, if there is any s_i that deviates from the expected behaviour, then s_i is marked suspicious. As a result, it is the attackers who have to learn how to guess the legitimate behaviour; rather than the network security analysts learning novel attacks implemented by the attackers. Moreover, because of the dependency of information hidden to attackers, the anomaly solutions should be able to detect novel attacks. However, the anomaly solutions might generate higher false negative rate (i.e intrusion $\rightarrow$ normal), and the reason is twofold. Firstly, attackers might guess the legitimate pattern, and once it has happened, random or authorisation-based detection must be applied. Secondly, attackers might obtain access to the database with legitimate behaviour. In the worst-case scenario, attackers might inject their own patterns into the learning database.

To show a difference between the two approaches, suppose that for a misuse system, a set of detection rules is implemented, and should work in any setting (i.e. a black-box solution). Sometimes, these rules are referred to as signatures. Therefore, misuse detection system should be thought of as deterministic, rule-based solutions. Suppose there are three (misuse) data sets learned by IDS, and defined as follows:

$$
\begin{aligned}
R_m &= \{\text{any website in the Internet}\} \\
X_m &= \{\text{viruses.com,hackers.com,computerattacks.com}\} \\
Y_m &\approx \text{any website not in } X_m \\
&= R \backslash X
\end{aligned}
$$

Moreover, suppose that a request r_i can be of either $x \in X_m$ or $y \in Y_m$. A deterministic system could develop a detection rule $\delta(r_i)$, in the following way:

$$\delta(r_i) = \begin{cases} \text{report}, & \text{if } r_i = x \\ \text{ignore}, & \text{if } r_i \neq x \end{cases}$$

Therefore, one can argue that in general human beings should avoid visiting websites related to computer insecurity. Otherwise, an intrusion is reported. To obtain X a large database of web requests has been analysed, and an analyst has come up with theory that agent browsing any website encoded in X, should be reported. Therefore, for misuse systems - data collection denotes analysis of similar activities (e.g. a relation: website from $X_m \rightarrow$ suspicious behaviour), and implementing a decision over it. As it has been mentioned, this process is concentrated at attackers, and their behaviour. Therefore, the above rule is strongly dependent on attacker-like websites. Similarly, a threshold-based system (e.g. requested one page more than λ times in a time unit) are integral to misuse systems [119].

On the other hand, there are anomaly detection systems for which there is a preliminary data collection part. However, the sets with requests are presented as follows:

$$\begin{aligned} R_1 &= \{\text{any website in the Internet}\} \\ R_2 &= \{\text{any website visited by legitimate users}\} \\ X_a &= R_1 \backslash R_2 \\ Y_a &= R_2 \end{aligned}$$

Initially, a set of legitimate website has been trained (i.e. R_2). Subsequently, any website that has not been observed during the learning phase (i.e. $R_1 \backslash R_2$) is considered as anomalous. This approach assumes that there is a large database of legitimate requests. In addition, a very strong assumption of regularity of legitimate users is imposed as well. In other words, it is assumed that the size of the learning database should cover majority of legitimate behaviour.

Now, suppose that $r_i = hackers1.com$, and the intention of the requesting person is legitimate. If the misuse system is in use, the request will be processed (i.e $hackers1.com \notin X_m$). However, if the person intention are malicious the realisation of the request is harmful to the system. Therefore, it is preferable to use the anomalous system. As a result, regardless of the intention, the request is always blocked. Obviously, that prevents legitimate users from accessing malicious website (e.g. intrusion detection scientists). However, it is always possible to allow privi-

leged users access the inhibited website. In addition, from the security point of view, the risk related to a malicious person performing illegal activity is always grater to a privileged person changing intention from peaceful to harmful (i.e. the scientist turning into an attacker). Therefore, anomalous systems should be sought in the systems that are less intrusion-tolerant. Finally, observe that attackers might want to change the domain (e.g. hackers1.com, hackers2.com, hackers3.com, h4ckers1.com, h4ck3rs1.com, etc). Subsequently, unless the misuse system has been updated, only the anomalous system will suspend these domains.
In the subsequent section the survey is provided of current methods on unsolicited browsing.

Chapter 3

Literature Review

3.1 Approaches to Detect Unsolicited Browsing

As it has been noted in Section 1.2 the general process of intrusion detection should be divided into following sections: Data Collection, Feature Selection, Anomaly Measurement / Classification, Results analysis. The analysis of approaches starts with Feature Selection, and gives insight into the system assumptions on legitimate or attacking behaviour. As it is shown in the next section, some approaches model legitimate activity, while others focus on attacking patters. Furthermore, different ways of Data Collection are described. This varies from using data sets obtained at a website the authors have got access to, to obtaining freely available web traces. Finally, assumptions towards anomalous behaviour are described. However, it is presented that access to a trace with attacking activity is *not* available. As a result, attacking strategies are being *manually* generated, violating the assumptions of legitimate profile.

Research approaches related to unsolicited browsing are divided into the following categories:

1. **Learning systems** - these systems involve learning of either legitimate or malicious behaviour, and applying gained knowledge to perform classification. In general, they cannot be implemented as black-box solutions, for which training part is omitted. On the contrary, a training phase is required, in which a data set containing legitimate or attacking behaviour is processed, and patterns of behaviour selected. Otherwise, one can obtain a data set with labelled legitimate and malicious samples. However, this is only the case for crawler detection, as recorded DDoS attack traces against HTTP protocol are not common.

2. **Technical solutions** - these methods attempt to implement technical (i.e. deterministic) means, so-called technical fixes, to achieve near-optimal recognition rate. In general, this is achieved by system modification (e.g. by modification of a website code, or additional hardware introduction). In addition, rate-limiting (i.e. based on thresholds) methods are added into this category.

In the subsequent sections, the focus is given on two types of attacks: (i) DDoS attack caused by different types of flooding, and (ii) detection of malicious crawlers. Recall that (i) refers to the scenario in which a number of zombies try to overflow one of system servers. In principle, this might be a database server that has received a number of complicated queries, or a web server experiencing numerous agents downloading large files (say, films) from its repository. On the other hand, (ii) refers to different type of crawlers and their illegitimate activities. There are different types of crawlers, the most popular are bots that extract websites' content and store into Google's cache. However, there are undesired ones that are trying to extract email addresses for spam purposes or find out about existing prices of some goods (i.e. products, tickets, deals, etc.), to improve their owner's financial position on the market.

3.1.1 Related Work - Learning systems

The approaches towards detection of attacking hosts are dived further into two classes. The first class of methods strongly focuses on transitions among web objects or characteristics of packet arrival. In general, *sequences* of these entities provide classification information. Therefore, this group is referred to as **sequence-oriented** group. On the other hand, there is another branch of research in which sequences of protocol heuristics are transformed into vectors. Specifically for this group, HTTP-specific features are extracted, and used for classification (such as a number of HTML requests, or a number of times robots.txt file has been requested). The sequence of request or requests is still important, however, the stronger relevance is given to features that are protocol dependent, and thus the name, **protocol-oriented** group.

3.1.2 Learning systems - Sequence-Oriented Methods

In this section, different statistical techniques are surveyed for Sequence-Oriented methods. Initially, a description is given of chosen methods. Subsequently, the focus is put on the way that other research collect data of legitimate users. Next, a profile of malicious users is given.

Sequence-Oriented Methods - Brief Outline

Lu et al. [52] were the first ones to introduce Hidden Markov Models (HMM) into the field of HTTP-GET attack. HMM have found many application, predominately in speech recognition [68]. The authors of [52] have focused on difference between sending requests and downloading in-line objects. In principle, when clicking on a link, many additional web objects (e.g. images, dynamic scripts, etc.) are embedded into the requested page.

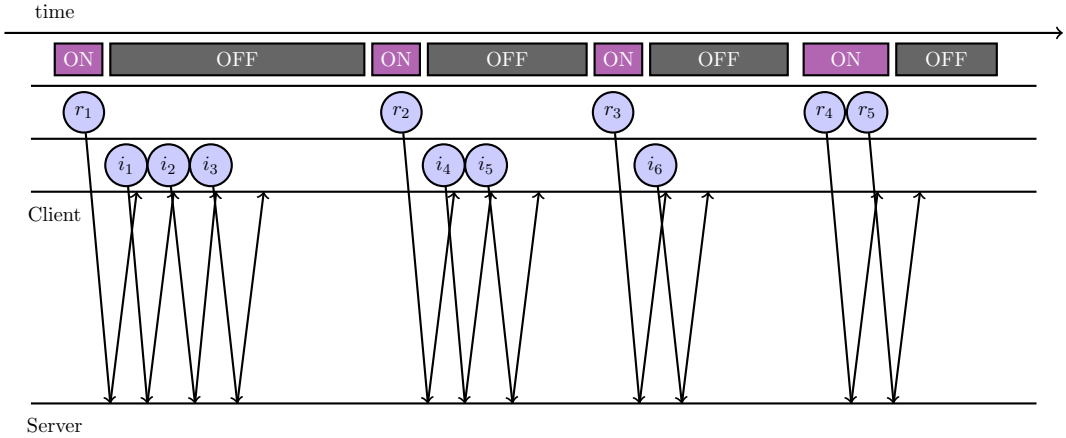

Figure 3.1: Transmission of requests (denoted with $r_{1:5}$) and in-line object requests (denoted with $i_{1:6}$) made by *legitimate* users. Observe that there are large time differences (marked with OFF) between two consecutive requests (i.e. activity time), and denoted with ON. However, time differences between two in-line object requests are fairly shorter.

This scenario is presented in Figure 3.1. From Figure 3.1 one can learn that there is a large time difference between two consecutive requests (denoted with r), most presumably for HTML objects. It seldom occurs that there are two requests in a very short time period (see r_4 and r_5 in Fig. 3.1). Note that the opposite holds for in-line object requests (denoted with i). Clearly, in-line objects are downloaded much faster, than normal requests. This behaviour is triggered by web servers that pipeline in-line objects, and transmit them as fast as possible. Moreover, inactivity time (referred to as OFF time), followed by sending a request is larger, comparing to time when legitimate users click. The OFF time is needed to process information (by legitimate agent) that has been obtained from the server, thus these periods are longer.

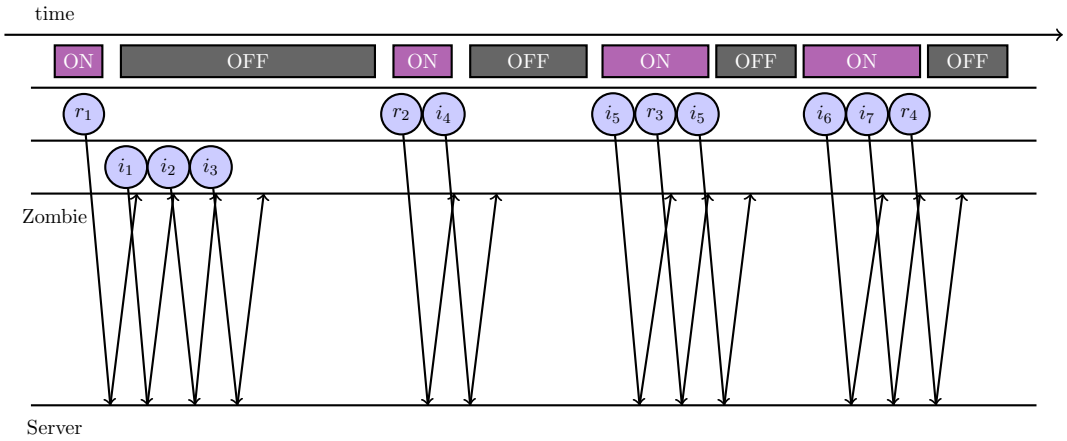

Figure 3.2: Transmission of requests (denoted with $r_{1:4}$) and in-line object requests (denoted with $i_{1:7}$) performed by *attacking* hosts. Observe that there are similar time differences between two consecutive requests, regardless of the type. Clearly, irregular object requests, and shorter inactivity time (marked with OFF), creates deviation from behaviour denoted in Fig. 3.1

However, transmission of zombies' requests do not follow this pattern. It stems from the fact that preprogrammed machines do not know which objects should be downloaded faster. Therefore, normal GET requests can be directed at objects that are normally in-line objects. Nevertheless, initially their transmission can follow some previously observed behaviour (see the beginning of transmission in Fig. 3.2). However, the aim of zombies is to overflow the server with requests. Therefore, their inactivity time (i.e. OFF time) is shorter. Moreover, during their ON time, zombies tend to request more objects, comparing to legitimate browsing. This difference is fundamental to many approaches that follow, and should be taken note of.

Moreover, it has been assumed in [52] that two-state transition matrix (i.e. downloading in-line objects or not) is informative enough for detection. Clearly, legitimate users should occupy the state representing OFF time much longer. However, the opposite holds for attacking hosts. Therefore, these hosts tend to change into the state responsible for ON time much often.

Subsequently, Xie has joined the co-author of [52], and extended the previously described approach. Moreover, the most popular branch of work on Application Layer attack based on HMM has been created by Xie et al. [98, 99, 100, 101, 102, 103]. Their work focused on two branches: detecting web proxy based attacks [98, 103], and detection of Application Layer attacks [99, 100, 101, 102]. Therefore, for this research [99, 100, 101, 102] are of main interest.

Fundamental to Xie's work [101, 102, 100] is the access matrix (also referred to as transition matrix), in which rows represent documents and columns correspond to time units. Each cell represents an average request rate of a web document (row index) in a time unit (column index), and each row is stochastic. Then, the authors have analysed a time depended Hidden Semi-Markov Model of transitions among web objects. Subsequently, in [100], and same as Lu in [53], they assumed that a legitimate request for a HTML link is followed by a number of in-line objects, embedded into that website. However, because zombies mostly download large or random objects [100], they fail to reproduce the actual time-dependent transitions (i.e. they frequently download the in-line objects that are unrelated and not present in the web log). Moreover, Xie has attempted to solve the problem of DDoS flooding attack.

Similarly to Xie, Lu et al. [53] have used HMM for detection of crawlers. Similarly, the authors of [53] have assumed that the difference between legitimate and attacking hosts is in the number of requests observed in a time unit. Crawlers generate requests at lower rate, same as human agents. However, once a human agent requested a web object, what follows, is the idle time in which the object is analysed; be it a text or image object. On the other hand, bot requests will arrive in more regular format, and the time gaps between two consecutive requests are shorter. Therefore, the above mentioned works are very similar in assumptions (e.g. transition matrix, HMM application, and formulation of malicious behaviour). Moreover, differences in both approaches are during training stage. Essentially, Xie et al. have trained their system on legitimate users, while the authors of [53] have used bot traces.

In the same manner, Yatagai et al. [106] have introduced legitimate profiles of legitimate behaviour based on time spent on a page. They have shown that there is a linear relation between amount of information and time a person reads a page (i.e. the more text or images on a page, the longer the page is read). Specifically, they have shown that there is a correlation between the number of objects, and inactivity time. This allows for detecting fast-requesting hosts that differ from the expected time scheme.

Another important statistical-based area of research against bots attacking HTTP websites uses information-theoretic measurements. These have been applied to intrusion detection problems numerous times. Lee et al. [47] have formally explained how to use entropy, conditional entropy, relative conditional entropy, information gain, and information cost in intrusion detection. Essentially, most authors have focused on relative entropy, often referred to as Kullback-Leibler (KL) diver-

gence [41]. It stems from the fact that KL divergence provides a way to measure distance between two probability distributions. Therefore, if one knows a legitimate distribution of packets (or requests) Q, then a distribution P of arriving packets (or requests) can be compared against Q to measure how different these two are.

This approach has been used by Wang et al. [94] to measure deviations of incoming *requests* at websites. Specifically, Wang observed that [94] KL divergence increases in parallel with arrival of new students (who do not know the website and randomly visit different categories). Subsequently, the authors have extended their work with the application of Large Deviations Theory [95].

Similarly, Yu et al. [110] have used KL divergence to measure deviation of incoming *packets* comparing to the legitimate profile of actual users. The authors have assumed a distributed system across more than one router. In addition, they have assumed that attacks should be generated using the same generating process, and therefore express similarity. As a result, by measuring arrival patterns across different routers, it is possible to discover regularities among attacking sequences. Yu has provided analysis of probability distributions metrics, and their performance against DDoS attacks [111]. Yu's research has subsequently focused on attacks against web servers, methods based on flows correlation [116] and packet padding [114] have been introduced. Moreover, Yu has presented how to mimic legitimate behaviour by obtaining partial access to a web-log of a website [117, 112]. Yu published a couple of papers introducing and motivating the research in the domain of application layer attack [115, 113]. Thapngam and Yu et al. [91] have extended Yu's previous work based on flow correlations.

KL divergence was also used by Ranjan et al. [70, 71, 72]. The authors have introduced workload profiles based on resource (i.e. server's CPU, database server's CPU, etc.) usage. Subsequently, they have clustered session based on similarity in resource consumption, given session requests. Therefore, KL divergence has been used to calculate deviations from the expected (optimal) resource used in a given profile.

The previously described works by Xie also utilised entropy to measure regularity of sequences [101, 102]. Session information is stored in normalised vectors denoting average interest in particular web object, in a time unit. Because zombies do not know legitimate transitions, the entropy of their vectors is higher, denoting less regular behaviour (comparing to the legitimate model). Moreover, it has been shown that for different values of standard deviation of entropy, the differences between legitimate and attacking samples of sequences are decisive.

Another statistical method has been introduced by Giralte et al. [28]. Their

method is a hybrid method that encompasses a technical solution (based on technical parameters of web sessions) and a graph theoretical representation of a website structure. In principle, the authors have analysed statistical parameters of transmission sessions (e.g. average number of packets sent in a session, average number of sessions per user, etc.). This way, attackers cannot overload website with only few infected hosts. In addition, they record transitions and minimum time between requested web objects. Therefore, bots cannot rapidly change among them. Moreover, they observe how often, a particular path has been generated. As a result, an observation of higher than average path iteration indicates suspicious behaviour.

Sequence-Oriented Methods - Data Collection

Data collection is a key process for IDS to perform successful classification. Recall there are two types of intrusion detection: misuse and anomaly as stated in Section 2.7. One of the differences between these two approaches is the process of data collection, depending on the preferred solution. Anomaly solutions focus on legitimate behaviour, while misuse systems try to build profiles of attacking hosts. So far, there has been no HTTP-GET attack recorded, and stored in a log format. Therefore, during training stage, researchers try to learn legitimate behaviour of the users, or parameters of the web servers during legitimate activity. Subsequently, attacking sequences of requests or packets are generated, based on different approaches described below. Observe that this does not hold for crawlers. Clearly, it is very common to observe crawler activity at website, and train IDS on their activity.

There are numerous different log format of web logs. However, the core format remains the same, and there are a set of attributes that remain prevalent. In general, these are listed below:

1. **Anonymised host ID** - as it has been mentioned (see Section 2.1), HTTP transmission takes place between valid hosts. This implies that both parties know their physical addresses. However, these are anonymised in order to provide anonymity to actual users who visited a website.

2. **Time-stamp** - exact time of the request

3. **Request** - Objects at websites can be of different types, for example: images (.img, .png, etc), text files (.txt, .pdf, etc), and videos among many others. Each file is identified by a uniform resource identifier (URI) of a requested object.

4. **HTTP reply code** - a server's numeric response to the request. Most common one is 200 and corresponds to "OK".

5. **Bytes in the reply** - this value denotes how much data (i.e. its actual size) was downloaded in the request.

As it has been noted in Section 2.7 anomaly based IDS's make a strong assumption about regularity among legitimate users that browse a website. This implies that users visiting a website will do so, in similar manner. Therefore, users browse alike pages by making matching transitions. As it has been pointed out in [101], there is a strong correlation among web flows (i.e. sequences of requests) stored in a log file containing user requests; sometimes referred to as traces. Independently, the same observation has been made by Jung et al. [35] for different HTTP traces. Therefore, most of the researches introduce this assumption, and analyse web logs to discover characteristic pattern of legitimate users.

Moreover, for academic purpose, a repository with HTTP traces has been created [1]. One of the popular data sets there is a log from an official FIFA Football World Cup 1998 site. It has been used by, Xie [101], Yu [116], and Thapngam [91]. In addition, there are many other data sets, and the other two frequently used for web mining techniques are CLARKNET [33, 86], and NASA data set [11, 60].

On the other hand, some scientists have obtained data sets from large organisations, in which the volume of traffic is large enough to build detection scheme. In addition to using World Cup 1998 data set, Xie [102] used a trace from a University log. Similarly, Yatagai et al. [106], Wang et al. [94, 95], Lu [53] and Lu et al. [52] have used traces obtained from their Universities. Giralte et al. [28] has used a trace from a e-commerce company.

Another approach taken by scientists is manual generation of legitimate behaviour. Yu used software-simulated legitimate connections with OPNET[110], and data set obtained from Massachusetts Institute of Technology and University of Waikato, [111]. Ranjan et al. [70, 71, 72] have used emulated legitimate behaviour, specified by a technical standard called TPC-W benchmark[2]. As a result, it is possible to generate arrival and request distributions that follow exponential distribution [71], with different parameters. As observed by Paxson, TCP arrival process follows this distribution [66]. Similarly, Lu [53] has assumed arrival of human agent requests that follow Poisson distribution. In addition, Lu is *the only* author that has attempted to learn behaviour of the attacking bots, and estimate likelihood of legitimate sequences on the model. Normally, this approach is common for

[1] `http://ita.ee.lbl.gov/html/traces.html`
[2] `http://www.tpc.org`

Protocol-Oriented approaches (see Section 3.2). Moreover, Yu [116] has used Pareto distribution of arrivals at websites. Parteo distribution has been used as a standard to provide distribution of interests varying from most popular to least popular web object [6].

Sequence-Oriented Methods - Malicious Behaviour

It is relatively easy to come up with a set of rules for misuse detection systems. Essentially, once an intrusion or virus infection has been spotted, a set of countermeasures is encoded into a database. This can be accomplished by blocking a particular host (i.e. by black-listing it), or updating file name in an anti-virus application.

However, an anomaly system cannot depend on predefined rules, and should focus on legitimate behaviour instead. As a result, initially the research community has decided to generate (i.e. prepare manually) a data set with legitimate and unauthorised network traffic [50], which has been developed at MIT Lincoln Laboratory. The growing popularity of the data set has drawn attention of data mining community, and in 1999 the data set has been chosen to be a "challenge" on the national 1999 KDD cup machine learning competition[3]. However, in the next years the creation process of the data set has been challenged, and critiqued in 2003 by Mahoney and Chan[55].

As a result, it has been a common practice to generate attacking sequences manually. Lu et al. [52] followed popular virus behaviour, and defined three different rates of attacks: increasing, decreasing, pulsing and constant. Moreover, following the virus description, the rate of requests is higher than normal users, making the attacking hosts remain in the state of obtaining in-line objects. Moreover, Yatagai et al. [106] have used the same virus definition, and the authors have assumed that attacking hosts follow increased rate of requests. Because their model adopts linear dependence between the amount of received information, and time difference before consecutive requests, the fast-requesting zombies deviate from the legitimate profile.

Analogous assumptions are introduced in Xie's work [98, 99, 100, 101, 103, 102] and state that the attackers would run the attack with increased frequency of requests. As a result, three different attacks have been devised: (i) Constant Rate Attack, (ii)Increasing Rate Attack, and (iii)Stochastic Pulsing Attacks. For (i) a zombie randomly sets request rate (that remains constant during transmission), duration of the attack, and delay from the global attack start. For (ii) the same parameters are used as in (i). However, zombie can decide when to increase the

[3] `http://kdd.ics.uci.edu/databases/kddcup99/task.html`

rate of requests, how long to keep it increased for, and when to decrease the rate. Therefore, the rate of (i) should be thought of as a linear and constant rate, while in (ii) it has a trapezoid-like shape. The most difficult attack to detect is (iii). For (iii), each parameter from (i) and (ii) is randomly generated. Therefore, different combinations of (decreasing/increasing) rate, duration and delay are considered for the attacking data set.

Similarly, Ranjan et al. [70, 71, 72] have introduced three types of attack: (i) Request Flooding Attack, (ii) Asymmetric Workload Attack, and (iii) Repeated One-Shot Attack. In (i), attacking hosts send requests at increased rate comparing to non-attacking nodes. Recall that Ranjan et al. have used different parameters of Poisson distribution to generate their legitimate and attacking flows. In (ii), the rate of requests generated by attacking hosts is not necessarily higher than legitimate requests. However, attackers have been informed about resource intensive requests, and their sessions are mostly filled up with them. In (iii), attackers take advantage of a large number of zombies, and each of the host is sending one very intensive request.

Yu et al. proposed three different attacking strategies in [110]: (i) fixed-rate attack, (ii) increasing rate attack, and (iii) attack based on varying parameters of Poisson distribution. Observe that (i) and (ii) correspond to the equivalently defined by Xie, while (iii) is similar to (i) and (ii) defined by Ranjan. In addition, a method of varying parameters of (iii) has been subsequently used in [111, 116], and extended by different parameters of Pareto distribution. Moreover, Yu has shown that it is possible to mimic legitimate behaviour by making bots follow Pareto distribution [117].

Thapngam and Yu et al. [91] have extended Yu's previous work based on flow correlations. Therefore, similar to Yu, Thapngam has introduced the following attacks: (i) increasing rate attack, (ii) periodical rate attack, (iii) non-predictable attack rate. Subsequently, it has been shown that by using Pearsons correlation coefficient, the authors can detect the three attacks.

Giralte et al. [28] assume four types of attacks: (i) random walk, (ii) login attack, (iii) search access and (iv) POST attack. These attacks imply that attackers make their zombies requests links or objects as they appear on page (i). In the other scenario (ii), zombies will generate a high number of invalid login attempts, or search requests (iii). These might overload a database server, by constantly sending database queries.

Reference	TP	FP
[52]	98.43%	6.6%
[100, 101, 102]	92.57%	7.5%
[53]	97.60%	0.2%
[106]	100.00%	10%

Table 3.1: Detection performance of Sequence-Oriented Methods

Sequence-Oriented Methods - Detection Performance

In general, results obtained in the sequence-oriented methods are reasonably good. Most popular way of providing documenting results is by showing the trade-off between TP rate (i.e. detection rate) and the rate of legitimate users misclassification (i.e. FP rate). These are presented in Table 3.1. Other researchers [28, 70, 71, 72, 116, 91, 94] do not provide the exact detection rates. On the other hand, detection results are plotted on charts. However, these plots indicate that the obtained results are very close to the ones presented in Table 3.1. Essentially, Ranjan et al. [70, 71, 72] have shown that 30 requests are needed to detect request flooding attack (very similar to attacking strategies defined in the coming Section 4.10). Wang et al. [94] have shown that entropies of legitimate users are very low, while random web browsing results in sessions generating high entropy. Yu et al. [116] have shown that there is a strong correlation between attacking flows, even though the attacks are delayed in time, and mixed with legitimate traffic.

3.1.3 Learning systems - Protocol-Oriented Methods

In this section, different techniques of IDS are described for Protocol-Oriented methods. This research group is mainly focused on detection of web crawlers that pretend to behave as legitimate agents. However, though in minority, there are approaches to detect flooding hosts.

Moreover, the protocol-oriented methods use features specific for HTTP protocol. These can be a number of error messages returned by server, a ratio of HTML to either images or document requests, or information embedded into a HTML request. Recall that Agobot uses popular web browser names or Google's bot ID's to hide zombies identity. Therefore, agents ID whether known or previously unseen can be used to improve classification. Therefore, features or vectors of features used in the subsequent section should be thought as composition of the above mentioned

parameters.

Initially, different approaches are described, followed by a set of attributes employed for recognition. Similarly, as before, this section concludes with the definition of malicious behaviour.

Protocol-Oriented Methods - Brief Outline

The motivation behind the research in this field has been brought by increasing statistics of crawlers activities at websites. As Tan et al. [88, 89] have pointed out, traffic generated by bots might vary daily between 5% and 10%. More recently Doran et al. [14] have observed that this volume has increased and ranges from 18.5% to 32.6%. Moreover, the total number of downloaded HTML object amounts to 85% [89]. In this research branch, majority of researchers extract session-related details into *vectors*, and train their classification systems with them. Observe that transition information are subsequently given less weight, and in most of the works, are not used.

For example, Stevanovic et al. [82, 84] have developed a detection based on Self Organising Maps (SOM) and Modified Adaptive Resonance Theory 2 (Modified ART2). In addition, the authors provided analyses of useful features to embed into detection vectors, and provided their importance for multiple classification algorithms [81, 83]. Similarly to [81, 83], Tan et al. [88, 89] analysed multiple features (described in detail in the next section) and used C4.5 model for classification.

Park et al. [64] have developed a system that measures mouse movement and keyboard strokes. However, the system requires software-dependent mechanism. In addition, they have tried to detect human agents by displaying a CAPTCHA (an acronym for "Completely Automated Public Turing test to tell Computers and Humans Apart"). CAPTCHA should be thought of as a "challenge" presented to a bot. Normally, it is a blurry image that conceals text message to be revoked by a human agent. Finally, they have used AdaBoost [74] to classify human agents and zombies.

Moreover, the authors of [31] have introduced their PUBCRAWL system, which initially builds a database of IP addresses of legitimate users and known bots (i.e. friendly bots, e.g. Googlebot). Their method is a hybrid method that combines the sequence-oriented traffic analyses, and support them with protocol-oriented heuristic. Moreover, when sequences are similar to both classes (i.e. legitimate and unlawful), their method embeds CAPTCHA challenges, described more in Section 3.1.4. Subsequently, the authors have tried to classify the arriving connections based on voting system using three classifiers: naive Bayes, associations

rules classifier and Support Vector Machine (SVM). The majority of votes from the three classifiers are to decide whether a session is coming from a crawler or a human agent.

Kang et al. [38] have concentrated their research at bots performing browsing and searching. In principle, it is assumed that actions that follow after search are crucial for bot detection. Moreover, the system has been developed that is based on a Bayesian network. They have built their network using spanning tree mechanism and iterative Expectation Maximisation algorithm to estimate required parameters.

To the contrary with the methods above, Lee et al. [45] have developed a sequence-independent method for flooding hosts taking part in DDoS attacks. Recall that this attack is mostly researched with sequence-oriented methods, described in Section 3.1.2. However, the authors have focused on sequence-independent attributes to build feature vectors. Subsequently, the vectors have been divided into clusters using K-mean clustering. Finally, for each cluster, a new feature representation is obtained with application of PCA, and selection of P-significant components. The error of estimation, calculated between initial vectors, and vectors obtained by PCA decomposition is assumed to be small for legitimate behaviour, and larger for bots.

Protocol-Oriented Methods - Data Collection

Once transitions among web objects are not considered, a number of features are introduced, which allow for intention classification. These heuristics are related to HTTP protocol parameters used in transmission among objects. There are numerous heuristics one can look at, but the most popular have been described in Table 3.2. Specifically, the features listed in Table 3.2, are common among other researchers. Also, features listed in Table 3.2 are stored in time-varying vectors compartments.

Feature Number	Attributed Name	Remark
1	Click rate	provides an average request rate of an agent
2	HTML-to-Image Ratio	the ratio allows for detection of agents that request too many HTML or image objects

Continued ...

Feature Number	Attributed Name	Remark
3	% of PDF/PS file requests	measures how popular PDF/PS requests are
4	% 4xx error responses	once a web object is requested with a GET request, the most common HTTP status code is 200, which simply denotes OK. Class 4xx errors denote situation in which request cannot be served, because of the user mistake. The most common error here is 404, denoting a request of non-existent object. However, in case of crawling bots 401 and 403 can often appear and represent unauthorised and forbidden access, respectively. The former requires authorisation, while the latter is returned whether or not, a user has been authorised.
5	% of HEAD requests	HEAD request behaves the same way as GET requests. However, HEAD requests are used more for diagnostic purpose, and returned data does not contain BODY part of a page (i.e. actual content). Therefore, it is recommended for bots to mostly generate HEAD requests [89].
6	% of request with unassigned referrers	this field displays information (commonly a HTTP address) about a page that referred (i.e had a link to) the requested resources. Crawlers are "encouraged" to leave out any referrer's information, and keep this field blank. Moreover, observe that historically this field is with a missing r in the middle (i.e. $referer$)
7	robots.txt request	in practice, each website should have robots.txt file in their top directory, and it should be requested by crawlers to obtain information about sections, permitted or forbidden for crawling. For detection purposes, this feature is binary and shows whether or not the file was requested in the beginning.

Continued ...

Feature Number	Attributed Name	Remark
8	number of HTML requests	this number should be relatively smaller comparing to other request (e.g. comparing to the number of requested images, or other in-line objects as observed in [100] and [53]). Observe that this feature denotes a number of different pages requested during session.
9	Total session time	the length of a session provided in seconds.
10	Average time delay between clicks	note that this feature is very similar to (1). However, when (1) gives an average number of requests, (10) measures average silence period between consecutive requests. Observe that same as in (8), (10) should be irregular for legitimate and more regular for attacking hosts [31].
11	Number of downloaded images	this feature should be useful against bots that tend to download large image files.
12	No of downloaded CGI scripts	CGI scripts are used to run commands at the server site. In general, legitimate browsing should not exceed a threshold on this parameter.
13	Unseen referrer	similar to (6), however, (13) denotes a referrer address that has not been seen before by IDS.
14	% of 3xx error responses	3xx errors inform user of additional action requirement. The most common error is 301 that warns of a permanently moved object. In general, this error appears when some objects have been moved, or their name changed. Bots might follow their own list of links that has been updated on the server. As a result, if their list is not renewed, subsequent request might result in many 3xx errors.

Continued ...

Feature Number	Attributed Name	Remark
15	Revisit rate	in general, legitimate users should revisit some popular content of a page quite often [31]. Therefore, low rate of (15) can indicate a malicious agent.
16	Accepting cookies	cookies have been introduced to store agent's past history. Moreover, these are very useful for hosts using address translation (see Section 2.5). It is assumed that bots should not have these feature implemented, which is not the case for legitimate users equipped with cookie-compatible web browsers [31].
17	Ordered requests	similar as feature (13), bots can obtain a website's structure, and share it with their crawling agents. These agents, can arrange links in an alphabetical(e.g. $\{news/objecta.html, news/objectb.html, weather/\}$) or a numerical order (e.g. $\{index.php?get = 1, index.php?get = 2, index.php?get = 10\}$)
18	Script Parameters	for dynamic websites, users are allowed to send different parameters to the server (e.g. $index.php?get = 1$ or $index.php?get = 1\&display = news$). Sometimes, these values can be difficult for bots to estimate, and randomly generated parameters might indicate malicious attempt.
19	Depth of traversal	different approaches use this feature in different ways. However, the most common approach is to denote (19) as the largest number of nested directories visited in a section (e.g. $/news/uk/london/$ denotes 3)
20	Number of Unique Searches	this feature is especially useful against searching crawlers that perform multi searches.

Table 3.2: HTTP protocol heuristics used as features in intrusion detection.

In this section $(\cdot)$ denotes application of a field, and $\neg(\cdot)$ denotes application of the opposite (e.g. (7) denotes opening of robots.txt during session, and $\neg$(7) denotes session without request of robots.txt file)

Stevanovic et al. [81, 82, 83] have used features: (1)-(7). Clearly, (1) has been used to detect zombies that request objects at increased rate. Features (2) and (3) assume legitimate interest regularity in specific files. In addition, the authors monitor error messages that one can come across during HTTP transmission (features (4) and (6)). Observe that (7) is characteristic for bots, and it follows a technical standard [89]. More recently Stevanovic et al.[84] have extended their feature set by introduction of path-traversal parameters: *wider* (15), and logarithm of (19). Wider revisiting should be thought of as visiting the same logical category, say */news/uk/london/* rather than */news/uk/london/news.html*. The authors of [81, 82, 84, 83] have used a trace obtained from their University.

Similarly to [81, 82, 84, 83], the authors of [88, 89] analysed features (1), (3), (4), (5), (6), (7), and (8), (19) in addition to 17 other features. Furthermore, the authors have used time-dependent features, namely (9) and (10), which have been selected to improve classification. The authors obtained a data set from their University, from which they have extracted required statistics on popular behaviour (e.g. known bots or known browsers).

Park et al. [64] have used data traces from their University network. For detection, except for the software-based solution, the authors have extracted similar features as the authors above. In principle, they have used (4), (5), (6), ¬(7), (11), (12), (13), (14) and other 4 less relevant parameters. The most useful fields in their system have turned out to be ¬(7), (13) and (14).

Jacob et al. [31] have defined two blocks for their detection. First block is heuristic based, as in the works described above, while second depends on time series analysis. In the heuristic block they have singled out the following features (4), (9), (7), increased rate of (8) coming from one agent, (12), (15), ¬(16), and (17). In addition, they have suggested to look at additional parameters, e.g. language selection that are often ignored by bots. In the time series block, the authors have focused on statistical analysis of transmitting request and autocorrelation.

Kang et al. [38] have observed that only 81.6% of web users have responded correctly to CAPTCHA challenges (described in detail in Section 3.1.4). Moreover, 99,9% of the agents *do not* respond to challenges at all. However, most of these agents who have refused to take on the challenge, are suspected to be bots. As a result, initially it is essential to select a group of agents that perceive some features, similar to bot behaviour. The authors of [38] have used (1), (8) and (20) when analysing behaviour of search agents. Some will send many search request but never visit any of them, while the other visit every returned object. Moreover, modification of (8) has been used to measure activity of agents including browsing

as well (i.e. performance not related to searching only), and referred to as total number of pages.

The authors of [45] have defined the following metrics: (i) a ratio of total number of requests seen in a session, to the average number of requests observed at a server, (ii) an interest vector, whose components denote relative frequency of request of a given web object, (iii) a fraction of requested unique web objects divided by the total number of web objects of the server, (iv) and highest frequency extracted from (ii). Therefore, (i), (iii), and (iv) are single-value attributes, and (ii) is a vector whose number of components is equal to the total number of web objects stored at a website. In addition, they authors have extended their vector representation by application of error messages (4), (14), and (18).

Protocol-Oriented Methods - Malicious Behaviour

Tan et al. [88, 89] defined few bots the authors have been interested in: (i) search engine robots, (ii) offline browsers, (iii) link checkers and (iv) email collectors. They have discovered that (i) will express broad width of traversal, but will have relatively shallow depth. It stems from the fact that search bots are oriented on the large number of files in short time period. That implies that (i) is interested in the main categories of a website, rather than deep-lying nested links. Similar differences will be observed for (ii). On the other hand, (iii) and (iv), have large numbers of unassigned referrer field. Moreover (iv) should have increased number of HEAD requests. In addition, the authors of [88, 89] expect bots to request robots.txt file. Moreover, the selected features number (5),(6)(7) and (8), have turned out to express positive correlation, when analysing attacking crawlers. Moreover, (8) and (11) have expressed strong anticorrelation, suggesting that crawlers will avoid images while browsing.

Similarly, Kang et al. [38] have observed difference between legitimate users, and searching bots. In principle, the authors of [38] have been able to discover that there is a very weak correlation between search results and media content (e.g. video or images). In principle, bots are more interested in text content (i.e. information that should be easier to process by text analysing machines), while human agents express similar interest in both text and media content.

As it has been pointed out, Park et al. [64] have made very strong software-reliable assumption. However, if it does not hold (i.e. human agents do not use JavaScript), then their experiments have been extended with additional statistics to detect crawlers. In principle, the authors have noticed there is a positive correlation between the downloads of CSS files (used for graphical design of websites) and

JavaScript files, during legitimate sessions. Therefore, the indication of malicious hosts is based on avoiding JavaScript and CSS files while browsing. Observe that this is similar to assumptions obtained in [88, 89], and in regards to image files. Moreover, the authors of [64] are the only group that have not considered depth of visit (i.e. feature (19) from Table 3.2), to detect crawlers.

Most recently, some of the assumptions described above have been reiterated by Jacob et al. [31]. First of all, revisiting is very common for human agents. This implies that the depth and width of crawler traversals are different to human agents. Clearly, crawlers perform extensive web crawl, rather than visiting popular paths. In addition, most of the heuristics should not be considered as they are easily faked, and only few of them should be considered (see previous Section). Moreover, recall that some sequence-oriented methods have assumed regularities in traffic pattern among bots, and inactivity time of legitimate users between requests. Similar observations have been drawn by Jacob et al. [31], when analysed traffic from a popular social website. In [31], the authors analysed autocorrelation of counting processes in terms of number of downloaded object in a time unit. Subsequently, they have shown that there are two different correlations. For legitimate traffic, there are similarities in a long range. Specifically, when looking at six hour-long and 12 hour-long time periods. This seems to indicate that after six hours of work, the users of the social website tend to login in, presumably during the lunch break. Similarly, the same process appears after 12 hours of work, likely while at home in the evening. On to contrary, the correlation among bots is observed for short time sequences, and decays linearly. Moreover, the authors have decomposed the sequences, and have shown that crawlers traffic is very stable, comparing to legitimate with spikes. As a result, same as in [100] and [53], the irregularity of legitimate users provides a detection feature.

On the contrary, Stevanovic et al. [81, 82, 83] have focused on the features listed in Table 3.2. Firstly, the authors have assumed that human agents will not request robots.txt file. The class labels are introduced as follows: (i) human visitors are flows during which robots.txt file was not requested. Otherwise, the flows are classified as either (ii) well-behaved or (iii) malicious crawler. Otherwise, sessions with unknown bot id are labelled as (iv) unknown visitors.

Recall that Lee et al. [45] are the only research group that have utilised sequence-independent parameters to detect flooding hosts. They have assumed three different attacking strategies: (i) main page, (ii) dominant page and (iii) random page attack. Their implemented strategies, target: (i) main site of a page, requesting it constantly; (ii) "popular pages", requesting them randomly, and a (iii) random

page from a set of all the pages. Moreover, (ii) strategy has turned out to generate lowest TP.

Protocol-Oriented Methods - Detection Performance

Similar, as for Sequence-Oriented methods, a table of popular rates has been obtained, containing results of detection performance

Reference	TP	FP	Recall	Precision
[82]	95%	4%		
[89]			83%	96%
[31]	95.58%	17.50%		
[38]			72%	75%
[45]	91.40%	9.1%		
[64]	99%	N \A		

Table 3.3: Detection performance of Protocol-Oriented Methods

Similarly, as for Sequence-Oriented methods, the results are depicted with performance metrics defined in Section 2.6. Observe that these systems generate slightly worse results comparing to the solutions that utilise transitions among objects.

3.1.4 Related Work - Technical Solutions

Clearly, the easiest method of web users authentication is the application of login and password combinations. Essentially, passwords are used even in sensitive services, such as on-line banking or e-shopping. However, while protection is required during risk-involved browsing, it should not be imposed at pages, where risk factor is not present. Moreover, security of databases containing passwords might easily fail [23]. All these reasons provide justification for password-free authentication detection methods. Previously, statistical and learning systems have been described. In this section, a survey is provided on technical solutions. Most of the methods described below modify the way that web servers process incoming connections. In addition, rate-limited and threshold-reliant methods are included in this category

Technical Solutions - Brief Outline

In the typical scenario of DDoS attack, a website is a single point of failure. In principle, attacking hosts keep on sending GET requests until the website is unable to service another user. Therefore, because it is difficult to deal with the high volume of traffic by application of a single website, one can come up with a solution where a frequently downloaded website's content (e.g. the in-line objects) is *replicated* onto multiple servers. Moreover, as it was explained by Wendell et al. [97], Amazon provides dynamic object distribution as well (e.g. cookie sharing). Decision of which server should be used is based on some allocation algorithm; most-commonly a physical location. Therefore, the closest server is elicited from the same country or continent. On the other hand, depending on the current network performance, another general approach is based on lowest latency [97]. This method is referred to as a content delivery network or content distribution network (CDN), and has been used by Jung et al. [35] and Chen et al. [10]. Clearly, this is a large-scale solutions that involves multiple server deployment across the globe, and implementation of transmission protocols (often from scratch), as reported by Wendell et al. [97].

More centralised approaches involve code-modification, and rarely utilise additional hardware upgrades. One of the earliest solutions to unsolicited browsing, were dynamically linked hidden objects. One of the methods described in Section 3.1.2 and developed by Park et al. [64] depended on hidden code to measure mouse and key stroke performance. Similarly, Gavrilis et al. [25, 24] has introduced methods based on dynamically adding invisible links. These so-called decoy links, are added into a page with the background colour or under images. As a result, it is virtually impossible to click the traps by a human agent operating with mouse. Observe that crawlers extract URI's of web objects, and send requests based on the returned set of HTTP addresses. Eventually, crawlers should request an object that is a trap, and trigger the alarm. Note that when using text-based web browsers (e.g. Lynx[4], which has been constantly developed and improved for more than 21 years now, and is still in use) hidden links are displayed. Therefore, a suspicious user might decide to use it. Moreover, Oikonomou et al. [62] have extended approaches presented in [25, 24]. Their invisible traps consisted of invisible images, text and forms, 1-by-1 pixel maps, and hidden objects underneath larger objects.

Note that the introduction of decoy links requires modification at the server site only. As a result, users do not need to modify their web browsers in order to access web resources. Moreover, there is no assumption about software compat-

[4] `http://lynx.isc.org/`

ibility (e.g. requirements of cookie-enabled application). On the contrary, some solutions require dynamic web language to provide protection (e.g. JavaScript), as in [64]. Identically, Srivatsa et al. [78, 79, 80] have introduced a two-layer system, performing user authentication, and providing rate-limiting control. The main motivation behind their work is a strong argument that traffic statistics cannot help to distinguish among intention of arriving connections. Clearly, if the attacking hosts are requesting objects at the rate below the allowable maximum, they will fit in the statistical pattern of legitimate users. Moreover, this assumption is followed in this thesis, with the implications drawn in the motivation section (see Section 3.2, for details).

As a result, the authors have decided to implement a transparent solution at the server site that requires modification of IP layer. Similar, as in [64], initially users are presented with a challenge to solve (e.g. CAPTCHA puzzle). Therefore, this user authentication method requires JavaScript-reliant browser. Once, the challenge has been solved correctly a system generates and injects 16-bit key into TCP packets. Therefore, IDS filters out users based on this authenticator (i.e. users who failed the challenge do not have authenticator present in their packets). In addition, the rate limiting component regulates the way that users utilise medium (e.g. remaining bandwidth).

Similarly, Doron et al. [15], introduced new architecture to attenuate DDoS attacks against web servers. The authors have introduced a system based on two queues for well-behaving, and malicious users. Their utilisation is similar to rate limiting solutions mentioned above. In general, users whose packets have been put into the slow queue, will get their bandwidth reduced further if they continue to exceed the expected number of downloaded object or requests sent. The authors of [15], and the same as in [100], [53] have observed that HTTP traffic follows the "ON-OFF" pattern (see Section 3.1.2 for details). Similar as before, the legitimate users will download a number of objects (ON), and subsequently remain inactive (OFF) to analyse obtained information. OFF time is modelled with exponential moving average. Obviously, long inactivity period is more desired for the system. Moreover, hosts with long average OFF time, are prioritised. Therefore, once they change to rapid OFF time, their sessions are not moved to the slow queue instantly. Rather, they are given some trust and can continue sending requests at increased rate. However, if the expected OFF time exceeds some threshold value, then transmission with the client will be rate capped. Similarly, a dynamic bandwidth limiter is imposed during ON period (i.e. during downloading of objects). Moreover, the authors have used SYN-Cookies [48], to defend against address spoofing. In addi-

tion, this is the only approach that concentrate the detection mainly on the weight of requests (i.e. number of bytes).

Similar to [15], Yu et al. [109] have used exponential decay of suspension time for blacklisted hosts. Moreover, in a similar manner, Yu et al. [109] have developed a system based on trust management to measure sessions anomaly. In principle, the authors of [109] have defined five different and time-varying trust factors. Specifically, there is (i) a short-term factor considering a short-history request rate, (ii) the long-term version of (i), (iii) cumulative distrust denoting the number of times a client has been penalised, (iv) mistrust related with number of times a client has switched from legitimate to malicious user, and (v) a overall trust based on the remaining four attributes. In addition to the trust factor, the system stores identity information, such as IP address to blacklist unwanted hosts. Their method uses cookie files to store the above mentioned information about clients and their trust factors. The authors of [109] are assuming (similarly to [78, 79, 80]) that during the attack, legitimate users should be granted access to the website. Therefore, based on the rate related trust factor, their system is able to modify allocated bandwidth to rate-violating users. Moreover, long-term trust factors allow for detection of attackers that have taken over machines, which previously have been classified as system friendly.

Finally, there are methods based on CAPTCHA using image-based challenges to authenticate web users. As pointed out by Basso et al. [4], there are different groups of CAPTCHA using challenges based on: (i) text, (ii) image, and (iii) sound. In general, for (i), a blurry or written with a mixed or twisted font text is presented. One of the oldest and most commonly applied solutions is Gimpy[5]. However, current text-based solutions can be solved by attackers, as reported by Mori et al. [58]. For (ii), an image is displayed containing an object, and user is asked to give the name of it. However, this method might produce some false positive because some object can be described by a number of synonyms (e.g. apple and fruit are correct description of an apple). The last method (iii) assumes that human agents are able to decode an audio message, since bots should not be able to do that. However good the approach is, it might not be applicable to some users with audible difficulties. Therefore, the motivation behind the work of [4] is the potential solutions of existing problems with (i), (ii) and (iii). As a result, a method has been introduced based on drag-and-drop images or definitions. In particular, users are asked to place images on text descriptors, or the other way round. Observe that it is extremely difficult to move objects that are put on top of each other, let alone group them according

[5] `http://http://www.captcha.net/captchas/gimpy/`

to their definitions. Especially, if a programmed machines is asked to do that.

Technical Solutions - Data Collection

Observe that for technical solutions data collection is not precisely defined. It stems from the fact that these solutions are deterministic, and provide close to perfect classification. Moreover, these solutions should be thought of as misuse detection. As a result, as soon as attackers have found a way around, the system is not protected. Specifically, for systems based on decoy links [24, 25] and [62], website administrators are not required to perform any activities, and their task is limited to code modification. However, Oikonomou et al. [62] have used real traces to define realistically attacking strategies, which do not deviate in temporal statistics from legitimate traffic. Identically, the drag-and-drop CAPTCHA method assumes inability of zombies to encrypt the challenges. However, for threshold based solutions, it is essential to measure traffic parameters to specify realistic threshold values. Therefore, the dual-queue system developed by Basso et al. [4] requires the learning time of half a million sessions to minimise FP (i.e. legitimate user $\rightarrow$ attacking attempt), and keep it around 1%.

Technical Solutions - Malicious Behaviour

Main approach of CDN [10, 35, 97] are the fact that attacking hosts are unable to discover which server will serve their request. Moreover, the volume of traffic is shared by a number of servers, and the attack is distributed.

Gavrilis et al. [24, 25] introduced three different attacking strategies: (i) same page attack, (ii) random page attack (constant request of a random page), and (iii) attack based on random walk (i.e. requesting random objects, as they appear at the requested pages). Recall that (i) and (ii) very similar definitions of attacking strategies, as in [45], which has been presented in Section 3.2. Moreover, the authors have observed that (i) and (ii) are easily detected with then implemented software solutions. Essentially, by slight modification of the server's code, the attacks are easily detected. However (iii) was missed by the software detectors. As it has been mentioned, Oikonomou et al. [62] used real traces to devise a realistic attacking model. As a result, attacking session length varies between 5 and 50 request, and the gap between two consecutive requests varies from 1/10 second to 1/60 second.

Srivatsa et al. [78, 79, 80] have assumed 0-think time, of attacking hosts. They have shown that assuming a competition between legitimate users and attacking hosts for a limited-bandwidth medium, their approach has shown improved

performance compared to other technical solutions (e.g. SYN-Cookie [12], see Section 2.2).

Doron et al. [15] have implemented three different attacking strategies: (i) simple flooding, (ii) high-burst slow attackers, and (iii) high-burst slow attackers. The easiest attack to detect is (i) that constantly requests random objects at increased rate, and occupies the slow bandwidth queue. On the other hand, zombies of (ii) type, introduce random portion of inactivity to (likely) be assigned to the fast queue. Moreover, these attacking zombies will know the maximum number of bytes observable at the server, and subsequently do not exceed it. The opposite applies to (iii), in which attacking hosts repeat multiple requests, followed by inactivity period, once the observable minimum portion of data is reached.

Yu et al. [109] have developed four attacking strategies: a constant rate attack without waiting for server response, a random rate attack with adaptation to a slow-responding server, and a stealthy attack, when a zombie gains trust from the server, and subsequently increases the rate to of requests.

Technical Solutions - Detection Performance

In general, technical approaches differ from the learning systems, as in most of the cases, no FP rate is assumed (i.e. there is no error of legitimate user classification). For example, Srivatsa et al. [78, 79, 80] and Doron et al. [15] presented their results as the trade-off between number of attacking hosts, and the number of successful legitimate web sessions. For the system with decoy links, Gavrilis et al. [24, 25] have shown the relation between the number of traps embedded in a website and probability of detection of a bot. For the similar system, Oikonomou et al. [62] have shown that 92% of zombies are detected after 24 requests and all of them after 48 requests.

3.1.5 Analysis of Previous Approaches

Observe that most of the methods described in Section 3.1.2 and Section 3.1.3 assume strong regularity of legitimate users. It often refers to the similar time spent on pages, or transitions among links. Moreover, detection methods against flooding hosts assume more frequent than legitimate requests rate. As a result, attacking hosts are detected by analysing temporal information between two consecutive GET requests. In addition, the authors of works described in Section 3.1.4 assume that due to numerous requests, zombies activate traps hidden on a page. Furthermore, these bots are unable to copy mouse and keyboard operation, nor can they guess

randomly-generated port of a multi-port web server [79, 78, 80]. Moreover, flooding hosts are unable to discover structure of CDN [10, 35, 97], and their distributed approach centralised at one victim, is split among multiple servers.

Similarly, detection of crawlers depends strongly on information encoded in HTML semantics, as has been described in Section 3.1.3. Essentially, the authors assume that crawlers include their identification information (e.g. do not leave agent field blank). Moreover, it has been shown that crawlers mostly ignore media content (e.g. images or video content), and never request them. Similarly, robots.txt file should be mostly requested by bots, and rarely by human agents. Moreover, there are long-term correlations (i.e. six and 12 hours) observed in traffic of legitimate users [31]. However, the authors very often acknowledge that it is difficult to detect crawling or flooding bots, and often tend to implement technical solutions based on CAPTCHA.

In the subsequent section a constructive critique of current methods is provided, which leads to formulation of a problem this research is tackling. Moreover, given the conclusions taken from others works, a set of attacking strategies are formulated that are difficult to detect by other methods.

3.2 Motivation for Research

In order to draw conclusion on the presented methods, a set of strategies are introduced, which attackers might take before running the attack against a web server. Initially, a website is thoroughly surveyed by the attackers. This process is divided into 3 main stages:

1. **Technical Recognition** - in this stage, a website is analysed to discover whether there are any technical mechanisms that could lead to bot detection. Essentially, this refers to any traps in a form of decoy links or objects.

2. **Semantic Recognition** - in this phase, attackers are building information on the website's structure. In particular, they perform random walk to discover implemented URI's. Observe that bots do not need to download images, view embedded films or documents. Rather, they can follow hyper-links as they appear at pages to pretend to be legitimate users, and store information about the embedded file and their types.

3. **Rate and Time Profiling** - in this phase, attackers attempt to discover the minimum, and maximum rate at which request are allowed to be made. In particular, it is assumed that the website operates some rate-limiting mechanism

against the frequently requesting zombies. Moreover, attackers are trying to estimate the allowable minimum and maximum time difference, between two consecutive requests.

Regarding (1), observe that current pattern recognition techniques are able to break visual CAPTCHA, as it has been shown by Yan et al. [104]. Moreover, website administrators might be reluctant to use these methods (i.e. either decoy links or CAPTCHA challenges), as these are software-reliant solutions. In addition, Lynx, one of the longest developed text-based browser (see Section 3.1.4), does not support JavaScript that is essential to run these methods. Therefore, methods based on keyboard or mouse motion detection fail as well. In addition, popular cookies applets might not be enabled in the user browsers. In addition, as it has been observed by Kang et al. [38], almost 100% of the web users are put off by solving CAPTCHA methods, and do not respond to them. Therefore, analysis of (1) can lead to conclusion that there is no technical prevention mechanism implemented at the website, because of the user-friendly approach of the administrators towards the website visitors.

Analysis of (2) allows for building of a structural model, as well as removal of unwanted links. Observe that a parsing bot can extract links leading to another page. If used, they can attenuate the attack, as the bots will effectively be redirected to another page. Implementation of (2) is essential for attackers, and as it has been reported by [35], during the DDoS attack attackers implement their zombies to request few links only. As a result, during (2), not only the removal of undesired links takes place. Moreover, attackers attempt to select potentially *popular* links to increase validity of their bots behaviour. Clearly, the more popular link has been requested by a zombie, the higher the chance the zombie will not be spotted.

During (3), attackers are trying to discover whether rate limiting IDS's are implemented on the site. Moreover, they are trying to discover the maximum time a bot is allowed to remain inactive for, before its reserved resources are released.

As a result of the above assumptions, attackers are happy to forfeit a number of zombies in order to find out time and technical constraints present at the website. This process is described with Algorithm 1. The *positive* parameter γ (in line 6 of Algorithm 1) specifies the speed of decreasing the rate at which requests are sent, to discover the lower bound of silence time between requests t_{min}. On the contrary, to estimate the upper bound t_{max}, attackers increment the value of r_i, by application of *negative* parameter γ. Therefore, attackers always sense a website first, before running the attack. In addition, attackers are able to discover hidden hyper-links or hyper-links leading to different pages, by simple content analysis. Observe that

Algorithm 1 Initial Recognition

launch a zombie z_i with initial request rate r_i, and initial page l_i
repeat
    parse l_n links from the current page l_i
    pick arbitrary l_j link $l_i = l_j$
    if z_i not blocked **then**
        request l_i after $r_i = r_i - \gamma$ seconds
    end if
until z_i blocked

the reason why a zombie has been blocked is either because it has requested the subsequent link too fast, too slow or has activated a trap. If the rate was within the legitimate range defined by attackers, the previous link must have been a trap.

To run the attack, attackers are assumed to implement their zombies to send requests at the legitimate rate, learned with Algorithm 1. The rate varies, is randomly chosen, such that a choice of time delay for next request follows uniform distribution on a sample space $\Omega = \{t_{min}; t_{max}\}$. Therefore, it remains within the border of the legitimate rate, as estimated with Algorithm 1.

Moreover, once the website has been chosen and scrutinised, and bots properly implemented, attackers perform the attack in a manner presented in Algorithm 2. It has been presented in Algorithm 2 that initially zombies randomly pick a starting page from the set of previously selected pages. Subsequently, the bots parse l_n available links, and remove any unwanted link that has been nominated by the bot master as invalid. Then, a randomly generated link is chosen from remaining links, and the whole process repeats. It is assumed that the time delay t_i in Algorithm 2 is within the acceptable range, discovered while performing the initial recognition (i.e. Algorithm 1).

Algorithm 2 The attack scenario

launch a set Z of zombies at initial starting page l_i (most likely, homepage)
for all $z_i \in Z$ **do**
    repeat
        parse l_n links from the current page l_i
        removed undesired links
        pick arbitrary link l_j, such that $l_i = l_j$
        if z_i not blocked **then** request l_i after $t_i = \mathcal{U}(t_{min}, t_{max})$ seconds
        end if
    until z_i blocked
end for

An example of one attacking session is presented in Fig. 3.3, where a snapshot of

the session is presented during which $\{\ldots, l_5, l_1, l_3, l_{10}\}$ have been requested. Suppose, for simplicity, that after each request r, there are always 10 links appearing at a page. Having run Algorithm 1, blue and violet links have been selected as the potential popular links. For example, after r_{n-2}-th request, there are 10 links, but only l_1, l_5, l_7 and l_{10} are promising for attackers. Subsequently, object l_1 is requested with some time delay equal to t_{n-2}, such that $t_{n-2} \in \{t_{min}, t_{max}\}$. Once r_{n-2} request has been made, there are new links for which, again, there are 10 new objects to request. The actual path of the zombie has been denoted with solid arrows, while potential transitions with dashed arrows.

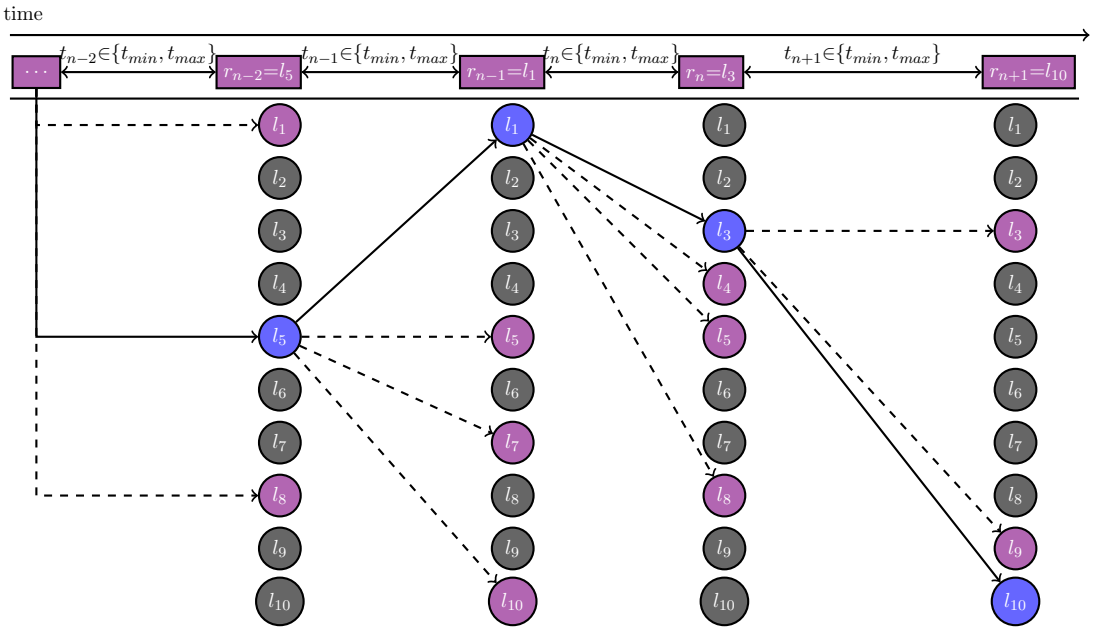

Figure 3.3: Once the structure of a website and requests rate boundaries have been learned (see Algorithm 1), zombies are able to avoid undesired links. These can be seemingly unpopular web objects, or traps planted by web administrators (marked with black colour). As a result, while making transitions bots are instructed to request extracted links (marked with blue and violet colour), and remain within time boundaries to make their presence legitimate.

Therefore, given the attack implementation in Algorithm 2, it seems possible to defend rate-dependent detection methods. Moreover, observe that behaviour presented in in Fig. 3.3, differs from the previously described approaches that assume frequently-requesting bots. Therefore, these methods will struggle with bots that follow the implementation of Algorithm 2. Moreover, recall that according to Listing 2.1, it is a common practice for attackers to hide their identity, by manual implementation of HTTP fields required by the protocol. Therefore, methods based

on protocol-related information (see Section 3.1.3) will find it difficult to detect attacking hosts that appear to be legitimate hosts.

Therefore, currently, the largest potential to deal with flooding DDoS zombies is given to attack-resistant methods. The motivation behind these methods is to distribute attacking traffic among multiple servers. As it has been described in Section 3.1.4, these methods are referred to as CDN [10, 35, 97]. In principle, once the increased number of users is observed, their requests are not directed to the web server. Rather, they are served by web proxies that should contain up-to-date version of the website's content. Observe that CDN *is* able to deal with the attack described in Algorithm 2. However, attackers might try and overload web proxies by running the attack from zombies that are physically close to the website (e.g. being placed in the same country). Moreover, CDN are commonly used to host *static* objects (e.g. in-line objects downloaded with a single request). However, hosting dynamic objects (e.g. PHP scripts) is challenging because of the dynamically changing result of each request. Indeed, most recently Wendell et al. [97] have reported that Amazon have implemented CDN including dynamic content distribution. However, as it has been reported in [97], estimation of time delays for CDN using dynamic content might decrease the performance of legitimate users. Therefore, implementation of dynamic elements into CDN is still an open research question.

However, in this thesis, it is assumed that *interest* can be used to measure *intention* of arriving hosts. In principle, suppose there is a way to *cluster* sequences of requests expressing similar interest in website's content. Specifically, suppose that users interested in *news* and *weather* could be different from users interested in *sport* ans *shopping*. This assumption holds when visitors from one group (i.e. browsing *news* and *weather*) avoid visiting categories popular in the other group, and vice versa. As a result, observe that programmed bots can follow hyper links as they appear at website; so they can request web objects at a lower rate. However, because the actual interest in existing categories is unknown to the attackers, their bots will fail to repeat transitions similar to a particular group of users (e.g. interested in *news* and *weather*). Therefore, initially it is required to group sequences from a trace using some similarity measure, which allows for similar interest elicitation, and cluster formation. Subsequently, another measure is required to assess validity of a connection in a cluster. This measure should be: (i) attack-strategy independent (i.e. should be able to detect different browsing strategies chosen by attackers), (ii) and attack-scale independent (i.e. regardless of the number of browsing bots, the attempt should be discovered, especially when this number is very small). Therefore,

in the next chapter, approaches are taken to explain methods allowing for grouping of sequences, and measuring their fit into clusters.

3.2.1 Summary

In this chapter an introduction has been given to the problem of intrusion detection. It has been shown that the problem is still unsolved, and actively researched. Moreover, different types of victims have been described, and application servers selected as the basis of further research. However, it has been argued that detection of attacks against application server is very challenging. Indeed, some researchers claim that *intention* is the only difference among connections [102]. In addition, because of stealthy appearance of the zombies, their activity becomes indistinguishable from legitimate data flow [80]. Moreover, detection mechanisms based on traffic characteristics become invalid [101]. Therefore, the research against unsolicited browsing is still ongoing; trying to provide more insight into strategies against attacking hosts.

Moreover, an attempt has been taken to compare approaches to detect malicious agents performing unsolicited browsing. The undesired bots can be divided into two groups: (i) flooding and (ii) crawling agents. In general, attacking hosts of type (i), attempt to exhaust server resources by sending multiple and
or resource-intensive requests. On the other hand, (ii) follow more legitimate pattern of resource consumption. However, their intention remain malignant, and their behaviour is often considered harmful.

Moreover, the research of unsolicited browsing is split into two groups: Learning Systems, and Technical Solutions. While the first group attempts to model either attacking or legitimate hosts, the other methods implement technical fixes ranging from simple code implementation to a global solution based on CDN. Learning Systems are divided further into sequences-oriented, and protocol-oriented methods. While the former utilised temporal information of web sessions, the later focused on semantics of HTTP protocol and protocol-specific configuration.

Finally, a scenario of the attack has been presented, having analysed a sample of Agobot, a very popular bot client. Given the code, it has been shown that protocol-reliant methods can struggle with detection of malicious bots. This inability of detection stems from manually setup of protocol-dependent attributes that were meant to provide classification. Moreover, simple recognition of website rate-limiting specification allows for outwitting sequence-dependent methods. Therefore, research on detection of unsolicited browsing is still progressing.

In the subsequent chapter, methodology of this research is given to detect different types of attacking hosts. Specifically, different methods are provided that

allow for identification of *interest* of legitimate users that remains unknown to attackers. Specifically, given the actual structure of a website, legitimate connections visit similar pages in groups. As a result, users interested in similar pages, avoid browsing other pages that again seem to be interesting to different groups of users. Thus, because the attackers do not know the actual interest in a website's content, it is unlikely that they will reproduce request sequences of actual (legitimate) users. Moreover, a set of attacking strategies is defined to closely resemble legitimate behaviour. As a result, additional methods are provided to discover the attack taking place at a website, and classification of web sessions into legitimate and attacking.

Chapter 4

Methodology

Having defined the research problem in the previous Chapters, and motivation behind the future research (see Section 3.2), a set of approaches is described to provide detection of unsolicited browsing. Generally, the application of IDS can be divided into two processes. Initially, IDS starts with discovery of a potential (i) attacking *attempt* (i.e. on-going attack). In principle, it is essential to recognise that the behaviour of currently browsing hosts is malicious. Subsequently, on a per-connection basis, (ii) intention classification should be performed.

Description of the methods attempting (i) and (ii) are presented in Fig. 4.1. Initially, data collection technique is described, in which different web traces are analysed. This allows for a division of web interest into two types: macro-interest and micro-interest. The former defines interest in web categories (i.e. more general approach), while the later denotes URI based interest (i.e. more detailed interest in actual web objects residing *inside* web categories). Observe that, once a website is "wider" (i.e. contains multiple categories, for example any e-commerce website or news portal), the approach based on the macro-interest might turn out to be sufficient. However, contemporary website tend to be "deeper", rather than wider (i.e. contain multiple links inside a group of very general, and limited categories). Moreover, it turns out that the focus cannot be put on every single URI of a website. As a result, a data preprocessing part is required for micro-interest approach.

There is a couple of key components that are overlapping for micro-interest and macro-interest approaches. However, these are described in a very general way (i.e. there is no reference to either of the approaches in order to avoid duplication of description). Therefore, this *might* hinder the processes of reading this chapter. As a result, a detailed description of the subsequent Sections is given, in order to shape the structure of the chapter.

Therefore, having analysed the data sets, an insight into technical details of the traces is given, and more specific information provided (e.g. a distribution of sequence length in the data sets).

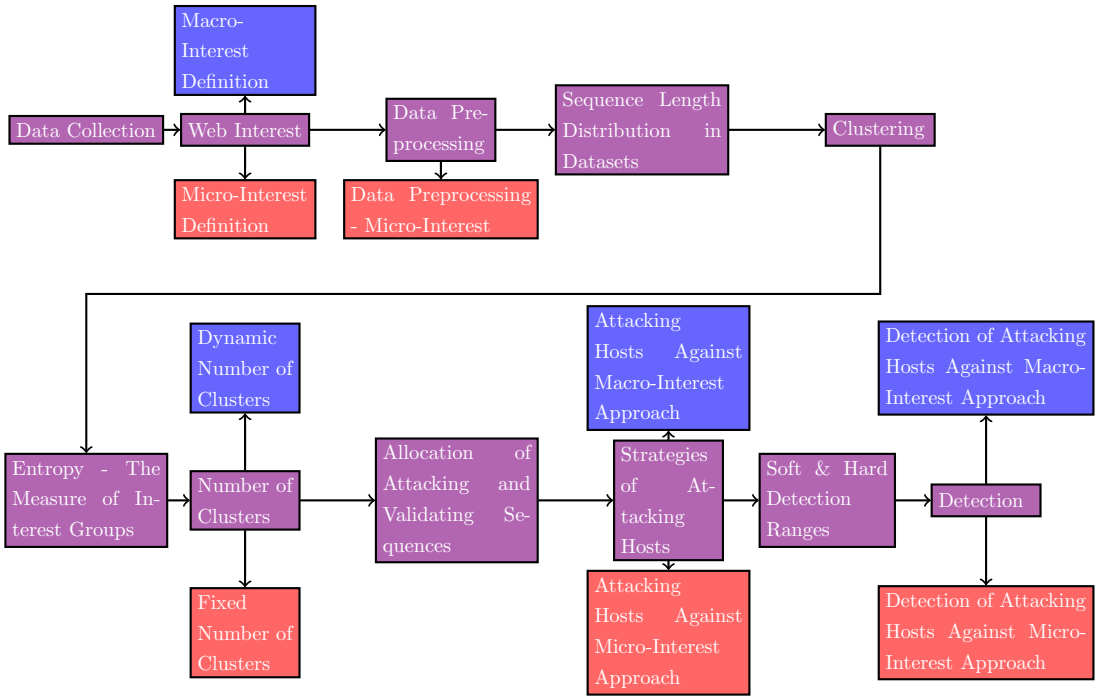

Figure 4.1: Organisation of this Chapter.

The above considerations lead to a definition of entropy-based clustering of sequences of requests. However, because the number of clusters cannot be defined in advance, two methods of cluster number specification are presented. Once legitimate sequences have been clustered, a method based on maximum likelihood is introduced that assigns validating and attacking sequences into the clusters. Interestingly, the distribution of the attacking sequences is indicative enough to provide (i). Recall that, most of the researches define their own sets of attacking strategies (see Section 3.1.2 for details). Clearly, those strategies challenge the assumption of the legitimate behaviour. Therefore, a set of attacking strategies is provided that, again, varies between macro-interest and micro-interest approaches. Moreover, clusters will be divided into ranges to apply different levels of detection techniques. As it has been noted, sequence distribution will allow for (i). Therefore, strict (i.e. "hard") detection techniques are imposed on the clusters mostly populated with attacking sequences. In contrast, detection thresholds are relaxed for the ("soft") clusters experiencing the surge of legitimate users. Once the division of clusters has

been accomplished, different statistical measurements are introduced to perform (ii).

4.1 Data Collection - Two Approaches to Web Interest

Before the clustering algorithm is presented, data sets obtained for clustering and validation of the approach are presented. The reason behind this, stems from the need to introduce the above-mentioned two approaches to measure user interest. Initially, interest is measured at a *macroscopic* scale, and interest in *web categories* is considered only. These can be: weather, news, sport, images etc. At this point, specific information on links, and objects inside each category *is not* considered. Therefore, suppose there are two requests inside *news* category. As a result, whether a nested link from *news* category, or a link pointing to *news* category has been requested, information stored inside a log about both actions will be stored as the categorical requests (i.e. *news*, followed by *news*). On the contrary, measured interest can be more detailed, and centralised at each URI. This time, it is analysed at a *microscopic* scale, and information inside data log varies (e.g. *news* and *news/sport/football*). Initially, macro-interest is explained, followed by micro-interest of web users.

4.1.1 Data Collection - Macro-Interest of Web Users

To give more insight into this approach, a freely available[1] data set has been obtained from a day-long activity of a web server, recorded at then location `http://msnbc.com`, on the 28^{th} of September, 1999 . The data set has been used before in detection of web interest [32],[77] and clustering of web sessions [42]. This data set is composed of sequences of numbers, corresponding to *categories* that were visited by users. During recording of the data set, there were 17 logical categories: *(1)frontpage, (2)news, (3)tech, (4)local, (5)opinion, (6)on-air, (7)misc, (8)weather, (9)msn-news, (10)health, (11)living, (12)business, (13)msn-sports, (14)sports, (15)summary, (16)bbs, (17)travel.* Moreover, the number of URLs inside each category varies from 10 to 5000. The sequences inside the data set are stored in the following format:

[1] `http://www.http://archive.ics.uci.edu/ml/machine-learning-databases/msnbc-mld/`

Listing 4.1: MSNBC data set format

```
1  Label format:
2  6 9 4 4 4 10 3 10 5 10 4 4 4
3  1 1 1 11 1 1 1
4  12 12
5  1 1
6  Text format:
7  on-air msn-news local local local health tech health opinion health
       local local local
8  frontpage frontpage frontpage living frontpage frontpage frontpage
9  business business
10 frontpage frontpage
```

From Listing 4.1, one can learn that only categorical requests are stored inside MSNBC data set. Moreover, observe there is no time information on when a request was made. Specifically, during first session, six different categories have been viewed in the 13-request long session: on-air, msn-news, local, health, tech, opinion (lines 2 and 7 from Listing 4.1). Specific information about requested link inside each category was not given. Therefore, as it has been mentioned above, only categorical interest is focused at. Moreover, a *depth* of each visit is not stored either. Therefore, most probably second session (lines 3 and 8 from Listing 4.1) visited category $frontpage$ "deeper", comparing to the fourth (lines 5 and 10 from Listing 4.1) session; *only* because of the larger number of $frontpage$ requests during the second session. However, this assumption cannot be validated.

4.1.2 Data Collection - Micro-Interest of Web Users

Suppose, there is a narrower version of the MSNBC website, and presented in Fig. 4.2. At any page, Top Menu is always present to allow for navigation among categories. Once a request has been made, a sub menu appears from the requested category. For the website presented in Fig. 4.2, there are five categories only, and each containing from 2 to 1,999 links. Observe that, currently, web pages are deeper rather than wider. As a result, the number of logical categories is smaller comparing to the total number of links. Clearly, most of the links of website presented in Fig. 4.2 are stored in the sub categories, making some categories densely populated. Subsequently, one wants to discover connections that expressed similar interest in links or web objects of any type (i.e links, images, documents, etc.), while browsing the website.

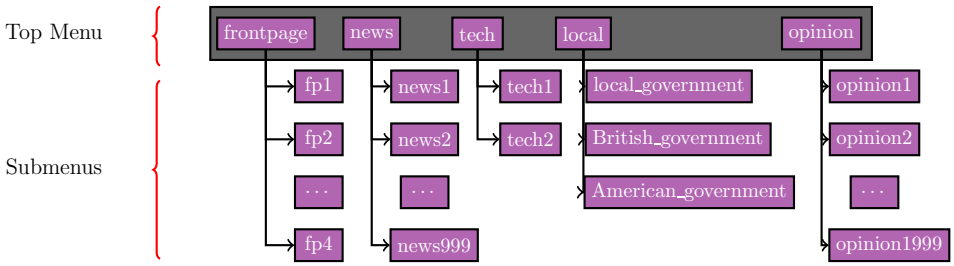

Figure 4.2: This Figure depicts a narrower version of the website presented in Listing 4.1. Observe that the number of existing web objects is relatively larger than logical categories.

Therefore, suppose that there are four interest groups (i.e. clusters consisting of frequently requested web objects) created by users interested in the following links:

1. $C_1 = \{fp1, tech2, local\_government\}$
2. $C_2 = \{fp4, tech2, local\_government\}$
3. $C_3 = \{news234, American\_government, local\_government\}$
4. $C_4 = \{news234, American\_government, opinion999\}$

If macro-interest was considered, first two groups would be merged, and represented by the same categorical representation i.e.: $C_{macro} = \{frontpage, tech, local\}$. However, because C_1 and C_2 appear in the log frequently and independently of each other, they should be treated as different interest groups. In addition, observe that it is more difficult for attackers to devise micro-interest group listed above (i.e. C_{1-4}). Attackers might assume that given the small number of existing web categories (i.e. 5) there is a specific concentration of users expressed with C_{macro}. Moreover, this information might be guessed with simple brute-force approach (described in detail in Algorithm 1, in Section 3.2). However, it is assumed that to figure out the actual micro-interest of C_{1-4}, access to the actual log recorded at a website must be granted. Therefore, attackers find micro-interest representation more challenging to break. However, macro-interest representation provides good insight into attacking strategies, what will be shown in the next chapter.

To cluster, and scrutinise the micro-interest approach, three data sets have been obtained, each consisting of sequences of users' requests. Specifically, as it has been presented in Section 3.1.2, users ID, exact time information, and requested object are provided. As a result, it is possible to devise an actual traversal path of each user. Two of the obtained data sets are available on-line[2], and are referred to

[2]http://ita.ee.lbl.gov/html/traces.html

as CLARKNET and NASA. CLARKENT spans two weeks (in August-September, 1995) of activity, and contains 3,328,587 requests. Inside NASA data set, in the same period (in July-August, 1995), 3,461,612 requests have been recorded. Access to the third data set, referred to as ESHOP, has been kindly provided by the authors of [28]. Inside ESHOP data set, there are only 64,044 different sessions observed during one day in 2009. The data set has been obtained to validate the proposed approach for a relatively *smaller* and *newer* data set. CLARKNET [33, 87] and NASA [11, 59, 105] have most recently been used in research related to different aspects of the behaviour of web users' modelling. Moreover, detailed description of these data sets should be sought in the relevant literature [2],[28].

4.2 Data Preprocessing - Micro-Interest

Each data set is divided into two subsets for training, and validating purposes. Subsequently, all unique objects and links requested during training are stored inside a data set L. Subsequently, *most popular* elements from L, are stored inside Ω. For the time being, assume that Ω contains *all* existing links from L. However, it is assumed that objects inside Ω should consist of popular elements, whose frequency of requests during training should be higher than some threshold λ. The reason behind doing this is twofold. *Firstly*, note that current websites can contain multiple links, causing large overhead. Moreover, as it is explained in the following section, each cluster is described by a joint distribution whose sample space is defined on a product $\Omega \times \Omega$. Clearly, for a system with multiple clusters, multiple instances of large Ω can produce too much overhead. *Secondly*, it has been proven by Kantardzic et al. [39] that only 10% of popular objects stored at a website are frequently requested. The implications drawn by the authors of [39] have been subsequently used in the previously described works of Wang et al [94, 95], and Xie et al. [100, 101, 102]. Moreover, recall that during flash crowd events the legitimate users visit popular links (see Section 2.5 for details). Nevertheless, if frequency of an object requests is below λ, transitions from and to this object are rare, and not informative. Therefore, λ is set to $\lambda = 0.1\%$. As a result, Ω consists of requests whose frequency is at least 0.1% inside training data set. It has been empirically tested that values below λ cannot provide enough information, for the detection techniques presented in the subsequent sections.

Afterwards, each sequence of requests is transformed into a sequence of numerical *labels* corresponding to indices of objects stored inside Ω. However, as it has been noticed, $\Omega \subseteq L$, and therefore objects that belong to a set $L \setminus \Omega$ should be

accounted for. If a request is unpopular, and not present in Ω, a metric based on Jaccard distance measure is used to find most similar object inside Ω.

Jaccard coefficient between two arbitrary sets A, B is calculated in the following way:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}, \tag{4.1}$$

and denotes a ratio of the number of the same elements belonging to A and B (i.e. intersection $|A \cap B|$) to all elements containing in A and B (i.e. union $|A \cup B|$). Jaccard distance is defined as

$$J_d(A,B) = 1 - J(A,B) \tag{4.2}$$

It has been shown that (4.2) is a proper metric [44]. For web domain, elements inside A and B could be thought of as sets consisting of elements of a single URI (e.g. for a link *website.com/news/sport* this data set consists of $\{website, news, sport\}$). In addition, we allow for existence of "others" category (i.e. index $|\Omega| + 1$) for which Jaccard distance is greater than $\frac{1}{2}$. This threshold is used to single out requests whose URI'es cannot find an object inside Ω for which at least 50% of URI'es components are the same. This value has been taken from literature [30], and is often used as metric of stability between two clusters. Moreover, suppose there is a unpopular link l_u, and there are two links l_i, l_j, such that $1 \leq i, j \leq, l_{n+1}, i \neq j$ for which $J_d(l_u, l_j) = J_d(l_u, l_i)$. In this scenario, a decision is made to select a link with the *lowest* popularity, to represent l_u. Clearly, that supports the idea that the object unobserved inside Ω should be represented by the object with the similar-looking URI, and low popularity percentage.

Recall the narrower version of MSNBC website presented in Fig. 4.2. Observe that there are $5+4+999+2+3+1999 = 3012$ observable links at the object (i.e 5 for the Top Menu, and 3007 in the sub categories). Subsequently, suppose that inside Ω there are 3010 links, excluding *tech*1, and *opinion*500. Subsequently, suppose there is the following session $s_i = \{tech, tech1, opinion, opinion500, Polish\_government\}$. Clearly, indices of *tech*, and *opinion* are inside Ω. However, the second, fourth and fifth requests are missing. Subsequently, *tech*2 is chosen to represent *tech*1 and *opinion*50 is chosen to represent *opinion*500. Observe that $J_d(tech1, tech\_) = J_d(tech1, tech2)$ (with _ denoting a white space), however, because *tech*2 is not residing in the top menu, its popularity factor is much lower than *tech*. Assuming that *opinion*50 obtains the lowest popularity factor during training, it is chosen over other similar links (e.g. *opinion*501). Moreover, *Polish_government* is represented as *British_government*, as its Jaccard distance is the smallest.

Therefore, observe that using (4.2) introduces labelling of unpopular links. Unpopular links are replaced with more popular ones, and requests that are not inside Ω, are represented by most similar objects. However, observe that this occurs for links whose frequency is very low, and should not introduce much noise into data. Therefore, objects inside Ω should be thought of as "centroids" of new "virtual" categories, observed at a website. Subsequently, the concentration of web users across these centroids (i.e. different combination of visits to the popular centroids) allows for interest group creation. Moreover, the clustering is performed *after* the sequences have been transformed into the corresponding indices (i.e. clustering is performed using indices of the categories). Similarly, sequences of requests coming from validating sequences are transformed into sequences of labels stored in Ω. Moreover, it is assumed that legitimate sequences make up for the training data sets. This way, the effect of missing replacements is minimised, and results in the subsequent sessions indicate that it can be ignored.

Moreover, recall that $\lambda_{links} = 0.1\%$. Similarly, Chen et al [9] divided web sessions into four groups, depending on the number of popular links inside each session, in the following way:

1. Grade 3, for which $10\% < l_p \leq 100$
2. Grade 2, for which $1\% < l_p \leq 10\%$
3. Grade 1, for which $0.1\% < l_p \leq 1\%$
4. Grade 0, for which $0\% \leq l_p \leq 0.1\%$

where l_p denotes link popularity. Moreover, object with the highest popularity rank will have popularity equal to 100%, and popularity of the other objects are obtained by division of the number of access of the most popular object. The authors of [9] analysed the World Cup 1998 data set (used frequently by other researchers, see Section 3.1.2 for details), and have drawn the following conclusion (regarding Grade 0):

1. Only 1.3% of users start browsing from the Grade 0 link
2. The average popularity of Grade 0 visitors does not change with the number of requests. Therefore, Grade 0 users will remain inside unpopular links.

Moreover, recall that during flash crowds, legitimate users follow the interest from the past history, and their interest follows Pareto distribution (see Section 2.5). Therefore, legitimate users will follow popular links and their request are directed

mostly at the most popular links, and thus, objects stored inside Ω. This implies that representing sequences of actual requests by labelling with the most popular URI's should be valid during a flash crowd event. As a result, requests generated by bots will rarely hit popular objects, and end up requesting objects denoted with index $|\Omega| + 1$. The same applies to popular crawlers that visit unrelated pages.

4.3 Sequence Length Distribution in Data Sets

Inside the data sets, most of the users would leave a website after a few requests, what has been presented in Fig 4.3. Clearly, sequences consisting of 30 requests are rare for ESHOP and MSNBC data set. Therefore, the more requests users make, the less likely the sequences appear in data sets. It has been assumed that to learn properly users behaviour depending on the length, it is required that sequences of different lengths should appear more than 1% inside data sets. Therefore, the maximum number of requests is assumed to be 30 for CLARKNET and NASA data set, while it is 18 for ESHOP, and 16 for MSNBC. In addition, observe that 30 requests

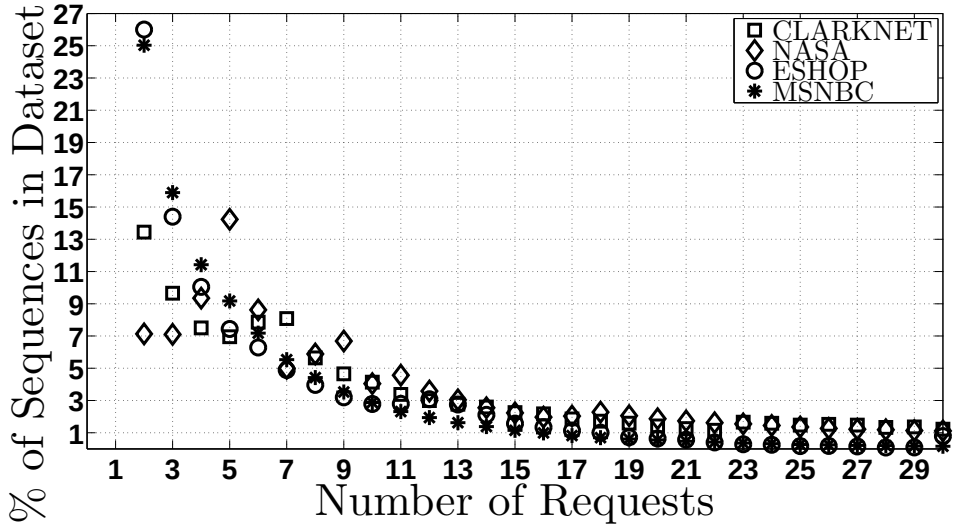

Figure 4.3: Length distribution of sequences inside traces.

for NASA and CLARKNET data set is slightly above 1%, and longer sessions could be used. However, Ranjan et al. [70, 71, 72] have estimated that 30 requests are required to learn request flooding attack (see Section 3.1.2 for details), and this limit is subsequently imposed in this thesis. Moreover, observe that for larger data sets users will have multiple requests spanned over two weeks (frequent users come and go, and their requests are identified with the same user ID, presumably IP address).

It has been assumed that the standard time-out for a session is 1800 seconds (30 min), as reported by Pitvko [67]. Identically, the time limit of 30 mins has been used by Stevanovic et al. [82, 84], Lu et al. [52] Jacob et al. [31], and Tan et al. [89]. Interestingly, this standard was introduced in 1996 [67], and has been in use ever since.

4.4 Clustering

Clustering algorithm described in this section is a general approach, and applied to the macro-interest and micro-interest methods explained in Sections 4.1.1 and 4.1.2, respectively.

Suppose there is a data set D containing n i.i.d. (independently and identically distributed) sequences of requests $s_1, s_2, \ldots, s_n$. Similar to the previous research approaches, it is assumed that D is composed of legitimate sequences. Moreover, it is assumed that regularity of legitimate user requests is very consistent, and allows for drawing conclusions on the legitimate behaviour. Subsequently, the data set is randomly divided into two sets: T for training (i.e. clustering), and V for validating containing n_T and n_V sequences, respectively. Set T is composed of roughly three times as many sequences as V. Moreover, attacking sessions are stored in sets marked with $A_{(\cdot)}$, where $(\cdot)$ denotes different attacking strategies. During validation phase, both sets V and $A_{(\cdot)}$ are merged, so that $VA_{(\cdot)} = V \cup A_{(\cdot)}$, containing n_{VA} numbers of sequences.

The aim of clustering C is to partition T into k clusters $C_i, 1 \leq i \leq k$, such that sequences in one cluster are similar to each other, and are different comparing to sequences in set $\{C \setminus C_i\}$. Moreover, for partition C the following two axioms hold:

1. T is partitioned k into disjoint subsets, such that:

$$C = \bigcup_{i=1}^{k} C_i, \;\; \forall C_i \neq \varnothing \tag{4.3}$$

2. Union of any two subsets does not result in an empty set:

$$C_i \cup C_j = \varnothing, 1 \leq i, j \leq k, i \neq j \tag{4.4}$$

Recall that, for the binary intention, IDS uses a classification function $f : T \rightarrow \{-1, +1\}$ that assigns a label $y = +1$ to potential legitimate sequences, and $y = -1$ is assigned to malicious sequences. Moreover, the length of sequences observed in D

varies between $r_\wedge$ and $r_\vee$. Thus, $r_\wedge$ and $r_\vee$ denote minimum and maximum number of requests, respectively. Therefore, given a sequence s_i, IDS tries and assigns the class $y_i \in \{-1, +1\}$, to the input sequences. Moreover, suppose that beforehand, IDS is given a *labelled* set of pairs $\{(s_1, y_1), (s_2, y_2), \ldots, (s_{|T|}, y_{|T|})\}$. In this scenario, IDS is required to group sequences into two clusters, such that $k = 2$. Moreover, IDS infers that there are some common features among $\{s_{1:|T|}\}$ that are explicit to legitimate sequences. Obviously, the opposite holds for attacking sequences. Therefore, the IDS can *learn* classification. Subsequently, given the input pairs, IDS tries to *estimate* the output values $\hat{y}$ of future sequences. This type of learning is called *supervised*.

The opposite to supervised learning is *unsupervised* learning, where the main difference is the *unknown* number of classes, an algorithm needs to assign data points to. Observe that, one can assume that for MSNBC data set, there are only 17 different interest groups, corresponding to the number of actual web categories. However, it is believed that a *combination* of the categorical interest (e.g. popular interest among 4 categories can be considered as one interest group, rather than four separate groups) should be sought after. Therefore, in general the number of clusters (i.e. interest groups) is *unknown*, and in the subsequent sessions it is shown how to estimate the number of clusters.

4.5 Entropy as a Measure of Interest Groups

As it has been mentioned before, a sample space $\Omega = \{\omega_1, \omega_2, \ldots, \omega_{n_C}\}$ is introduced, containing n_C numerical labels. Recall that this represents indices of popular links inside L (for micro-interest approach, described in detail in Section 4.1.2) *or* actual categories (for macro-interest approach, described in detail in Section 4.1.1) of a web server. Observe that in general, the number of web categories or extracted popular links stored in Ω, does *not* make up for the number of clusters, such that $n_C \neq k$. Recall that these are just labels, prepared for the purpose of observing an event from Ω.

For each $i, 1 \leq i \leq n$, s_i is a realisation of a random sequence that takes on values in Ω. Moreover, each sequence s_i consists of $n_i \in \{r_\wedge, \ldots, r_\vee\}$ numerical labels, such that $s_i = (s_i^1, s_i^2, \ldots, s_i^{n_i})$, and $r_\wedge, r_\vee$ denote minimum and maximum number of requests respectively. $r_\wedge$ is assumed to be 2, and $r_\vee$ has been set separately for different data sets (see Section 4.1.2 for details).

Furthermore, suppose that s_i is one of the longest sequences in T, such that

$n_i = r_\vee$. Subsequently, probability of such a sequence could be obtained, such that:

$$\begin{aligned} P(s_i) = P(s_i^1, s_i^2, \ldots, s_i^{n_i}) = P(s_i^{n_i} | s_i^1, \ldots, s_i^{n_i-1}) P(s_i^{n_i-2} | s_i^1, \ldots, s_i^{n_i-1}) \\ \ldots \quad P(s_i^2 | s_i^{n_1}) P(s_i^1) \end{aligned} \tag{4.5}$$

However, web traces are very limited, and in general it is impractical to obtain enough knowledge to learn (4.5) (see Section 4.3 for details). Moreover, because of memory overhead, large matrices corresponding to (4.5) are avoided as well. Therefore, suppose that (a, b) denotes observing a pair of requests, such that $(a, b) \in \Omega \times \Omega$. As a result, for the purpose of this research a bi-variate joint distribution

$$P_j(a, b) = \frac{N_{a,b}}{N} \tag{4.6}$$

is used to represent cluster C_j, where $j, 1 \leq j \leq k$.

To estimate joint distribution $P_j(a, b)$, the following terms are used:

$$\begin{aligned} N_{a,b} &\triangleq \text{ a number of times pair } (a, b) \text{ has been observed} \\ N_a &\triangleq \sum_{b \in \Omega} N_{a,b} = \text{ total number of transitions starting from } a\text{-th object} \\ N &\triangleq \sum_{a,b \in \Omega} N_{a,b} = \text{ total number of transitions among web objects} \end{aligned} \tag{4.7}$$

Therefore, given (4.7), a probability of observing a pair of requests (a, b) is approximated by (4.6), a marginal probability of observing request a is obtained by summation over each web object:

$$P(a) = \sum_{b \in \Omega} P(a, b) \triangleq \frac{N_a}{N} \tag{4.8}$$

Subsequently, given Bayes rule, a conditional probability of a given that b was the previous request is calculated in the following way:

$$P(a|b) = \frac{P(a, b)}{P(b)} \tag{4.9}$$

Moreover, a probability of sequence is obtained by applying product of (4.8) and (4.9), and is described in the subsequent sessions. Observe, that the assumption is made that the above frequency-based formulation can estimate transition probabilities, which follows the Weak Law of Large Numbers.

Moreover, this approach is often used in literature related to intrusion detection for websites. Recall that, Lu et al. [52] introduced system based on HMM

between two states defined as (i) requesting a hyper-link or (ii) an in-line objects. To define transitions between (i) and (ii), a table of counters is introduced, from which HMM can be estimated. Similarly, Xie et al.[99, 100, 101, 102] introduced a popularity matrix that is build from frequency of requests. Identical approaches are observed in different domain of web data mining. For example, Yang et al. [105] used similar approach when working with NASA data set, and trying to devise a pre-fetching algorithm.

For the random request a defined on Ω, entropy is calculated as [76]:

$$h(a) = -\sum_{a \in \Omega} P(\omega_i) \ln P(\omega_i) \tag{4.10}$$

In addition, entropy is bounded, such that

$$0 \leq h(a) \leq \ln(|\Omega|) \tag{4.11}$$

Specifically, (4.10) is equal to $\ln(|\Omega|)$, when (4.5) is uniform, which is sometimes referred to as a flat distribution.

One approach, for *categorical* data points clustering, utilises entropy minimisation [3, 49]. Entropy of (4.6) is calculated as [76]:

$$h(a,b) = -\sum_{a,b \in \Omega} P(a,b) \ln P(a,b) \tag{4.12}$$

Generally, high entropy corresponds to flatter distribution, whereas smaller entropy denotes skewer distribution. Recall that, Lee et al. [47] formalised usage of information-theoretic measurements for IDS. Thus, for IDS purposes one seeks clusters with smaller entropy to group elements representing similar behaviour [47]. Therefore, in order to group sessions stored in T into k clusters, k *empirical* joint distributions $P_{1:k}(a,b)$ are obtained by analysing sequences from each cluster. Specifically, initially each $P_j(a,b)$ is a uniform distribution from which each pair of requests can be drawn with equal probability equal to $\frac{1}{|\Omega| \times |\Omega|}$. Subsequently, $P_j(a,b)$ transforms into a more peaked distribution by application of (4.7).

Following Barbará [3], if there is set $C_{1:k} = \{C_1, C_2, \ldots, C_k\}$ of k clusters, the goal is to minimise the average entropy of the clusters, which is computed as follows:

$$\mathbb{E}\{h\} = \sum_{j=1}^{k} \left(\frac{|C_j|}{t+1} \{h_j\} \right) \tag{4.13}$$

Therefore, having grouped $t, 1 \leq t \leq n_T$ sequences from T given (4.13), every time a sequence $s_i, t < i \leq n_T$ is to be added to the composition of clusters $C_{1:k}$, a cluster $C_j, 1 \leq j \leq k$ is picked where assigning sequence s_i decreases the *scaled* (i.e. multiplied by $\frac{|C_j|}{t+1}$) cluster's entropy the most.

Note that clustering based on (4.13) attempts to solve two problems at once. Mutually, popularity of behaviour C_j is considered (i.e. the popularity term $\frac{|C_j|}{t+1}$ is calculated), together with minimisation of averaged entropy of cluster composition $C_{1:k}$. For sequential processing of data sets the following problems might arrive for high-dimensional data sets.

Observe that the difference in entropy Δh_j of cluster C_j before and after adding s_i, might be very small for data sets containing objects with many features. In addition, suppose there is a unpopular cluster $C_u, 1 \leq u \leq n_T, u \neq j$ for which $|C_j| \gg |C_u|$ (i.e. containing relatively fewer sequences comparing to C_j). Subsequently, suppose that s_i reduces entropy of both clusters. However, the reduction in entropy for the unpopular cluster is larger than for the popular one, so that $\Delta h_u > \Delta h_j$.

Now, the minimisation of (4.13) might lead to assigning a sequence to C_j over C_u. It stems from the fact that the popularity factor $\frac{|C_u|}{|t+1|}$ might decrease the high entropy improvement Δh_u. As a result, C_j might be absorbing sequences because of its high popularity factor.

Therefore, (4.13) is modified by introduction of the following minimisation function. While partitioning T, a cluster $C_i, 1 \leq i \leq k$ is chosen, for which introduction of currently processing sequence decreases the entropy of C_i the most, such that

$$i = \arg\max \Delta h_{1:k} \tag{4.14}$$

Finally, observe that cluster representation and minimisation of 4.14 allows for interest elicitation. Specifically, this unsupervised approach allows for representing the web interest as peaks of joint distributions. Moreover, sequences of requests are added to $C_{1:k}$ to increase the heights the of peaks, given (4.14).

4.6 Calculating the Number of Clusters

Recall that, initially, a clustering of sequences is sought to group sequences expressing similar interest in a website's content. Moreover, the number of different interest groups is unknown, and has to be estimated from data. For this purpose two approaches are taken. Firstly, a method based on dynamically changing k is presented. Subsequently, a method is introduced that utilises a fixed k.

4.6.1 Dynamically Changing the Number of Clusters

Observe that if k is not fixed, then sometimes creating a *new* cluster is actually the best placement for sequence s_i, given (4.14). Therefore, when k is not fixed, it changes while adding new sequences to $C_{1:k}$, and depends on the factors described below. Moreover, it is important to note that the order of sequences can play a crucial role in the clustering algorithm (i.e. decreasing or increasing its performance). In order to attenuate the unwanted effect of ordering, the following potentially improving processes have been introduced: Re-clustering, Merging and Partitioning.

Re-clustering

Re-clustering process is proposed to find a better cluster for a sequence $s_i, 1{\leq}i{\leq}b$, including clusters that were not present during initial addition of s_i, after having processed $b, 1{\leq}b{\leq}n_T$ sequences from T. In other words, suppose that while grouping s_i there are k clusters, and s_i has been added to $C_j, 1{\leq}j{\leq}k$. It might be the case, that after having grouped other b sequences, there is a different number of clusters k'. Subsequently, suppose there exists another cluster $C_{j'}, 1{\leq}j'{\leq}k', j' \neq j$ such that placing the previously added s_i inside $C_{j'}$, minimises (4.14) further. Therefore, after processing a batch of b sequences, the algorithm is stopped, and the whole set C is re-clustered, moving sequences to another cluster, if there is improvement in (4.14).

Merging

Yet another process that can minimise (4.14) is merging. While connections are being added to cluster composition C, it might be the case that there are some clusters that pose similar characteristics, however they exist as two different instances.

One way of comparing two joint distributions $P_i(a,b), 1{\leq}i{\leq}k$, and $P_j(a,b)$, $1{\leq}j{\leq}k, i{\neq}j$ is the application of Kullback-Leibler (KL) divergence [41] formula:

$$D_{KL}(P_i||P_j){=}\sum_{a,b\in\Omega}\sum P_i(a,b)\log\frac{P_i(a,b)}{P_j(a,b)} \tag{4.15}$$

However, KL divergence is not symmetric, nor does it satisfy triangle inequality. As a result, Jensen-Shannon divergence (JSD) is used, and calculated in the following way:

$$\begin{gathered} JSD(P_i||P_j) = \frac{1}{2}D_{KL}(P_i||M) + \frac{1}{2}D_{KL}(P_j||M), \\ \text{where} \quad M = \frac{1}{2}(P_i + P_j) \end{gathered} \tag{4.16}$$

and it has been shown by Endres, et al. [20], the square root of 4.16 is a proper metric. In addition, the metric is bounded, such that $0 \leq JSD(P_i||Q_i) \leq 1$. Thus, C_i and C_j are merged when $JSD(P_i||P_j) \leq \eta_{i,j}$, and $\eta_{i,j}$ is obtained in the following way:

$$\eta_{i,j} = \underset{\{C_{1:k} \setminus C_i\}}{\arg\min}\, JSD(P_i||P_{\{C_{1:k} \setminus C_i\}}) \tag{4.17}$$

where $\{C_{1:k} \setminus C_i\}$ denotes a set of all clusters except for cluster C_i

Partitioning

The clustering approach based on minimisation of (4.14) is *computationally expensive* [3]. In order to increase the speed of the algorithm, a partitioning of connections is introduced.

Before the algorithm is applied, data set T is partitioned into a set of partitions $\mathbb{B}$ containing a small number of partitions p. Each $B_i \in \mathbb{B}, 1 \leq i \leq p$ is a *range* of the ordered sequences' indexes, such that $B_i = \{B_i^1, \dots B_i^{d_i}\}$, where $1 \leq B_i^{d_i} \leq n_T$ denotes the index of a d_i-th sequence from T corresponding to the last element's index from B_i. For two adjacent ranges B_i, and $B_j = \{B_j^1, \dots, B_j^{d_j}\}, 1 \leq j \leq p, j \neq i$, the following relationship holds: $B_i^1 < B_i^{d_i} < B_j^1 < B_j^{d_j} \leq n_T$.

For each partition B_i, the clustering algorithm is applied with re-clustering and merging to prepare k_i clusters $C_i^B = \{C_i^1, C_i^2, \dots, C_i^{k_i}\}$ for block B_i. When each partition of indexes has been processed, each set of clusters C_i^B contains best arrangements of k_i clusters, given (4.14), for its own sample of connections *only*. Afterwards, the sets of clusters C_i^B are merged together. Subsequently, merging and re-clustering is performed on the newly obtained set of clusters $C = \{C_1^B, C_2^B, \dots, C_p^B\}$ until there is no improvement of (4.14). This process saves time, because merging and re-clustering are not as computationally expensive as minimising (4.14) for a large sample of connections, allowing k to dynamically change.

4.6.2 Summary of the Algorithm

In general, the clustering algorithm can be represented with Algorithm 3. Initially, connections are divided into p same-length blocks. For each block, 10 sequences are chosen to prepare 10 clusters. For each block, the clustering routine is performed to minimise its own version of (4.14). Merging and re-clustering are performed as well after every b connections have been processed within each $B_i, 1 \leq i \leq p$ block. When clustering of each block B_i has been finished and corresponding sets of clusters have been prepared, there are many clusters spread across composition $\mathbb{B}$. Thus, the

clusters from each B_i are gathered and analysed as one set of clusters C. Subsequently, until the value of (4.14) cannot be minimised further, set C is merged and re-clustered alternately.

Algorithm 3 Clustering algorithm

```
1: divide connections into p partitions 𝔹
2: for all B_i ∈ 𝔹, 1≤i≤p do
3:     j = B_i^11
4:     for all s_j, j ∈ B_i do
5:         no_of_processed_connections=10
6:         while no_of_processed_connections<b and j ≤ B_i^{d_i} do
7:             if adding s_j to C_i^t, 1≤t≤k_i minimises (4.14) then
8:                 assign s_j to C_i^t
9:             else
10:                n_k=k_i + 1, k_i=n_k
                   Create new cluster C_i^{n_k} containing s_j
11:            end if
12:            no_of_processed_connections++, j++
13:        end while
14:        Re-cluster B_i, Merge B_i
15:    end for
16: end for
17: integrate all C_{1:p}^{1:k} ∈ 𝔹 to get C
18: repeat
19:     Re-cluster C,Merge C
20: until (4.14) cannot be minimised further
```

4.6.3 Fixed Number of Clusters

The main reason why a fixed number of clusters is introduced, stems from the overhead the Algorithm 3 generated for high-dimensional data sets. Initially, the application of Algorithm 3 worked well for MSNBC data set. Clearly, because the number of web categories is relatively small (i.e. 17) the algorithm works fast enough to be implemented in real environment. However, because URI's (instead of web categories) are selected from the remaining data set, the cardinality of Ω is considerably higher. Thus, rather than changing k dynamically, the following method is proposed.

In this work, k is chosen to be $e^{h(a,b)}$ that is number of clusters needed for lossless compression [76]. $e^{h(a,b)}$ is obtained by analysing empirical joint distribution $\mathbf{P}(a,b)$ having analysed *all* connections belonging to T. It should be thought of as a joint distribution of all sequences from T, when there is only one cluster. Subsequently, once $k = e^{h(a,b)}$, k different sequences are selected that visit frequently popular transitions in $\mathbf{P}(a,b)$.

Having clustered set $C_{1:k}$, a set of unique patterns of interest is obtained. However, it is essential that set $C_{1:k}$ is reshuffled, and best placement for already clustered connections is searched for, based on further minimisation of (4.14). The main reason behind it is the fact that clustering based on entropy-minimisation detects many unique patterns that, not necessarily, are popular. Therefore, recall that when k dynamically changes, the sequences are reallocated as well (see Section 4.6.1). Similarly, reallocation is applied for the algorithm utilising the fixed k, and is described below.

4.6.4 Reallocation of Sequences for Micro-Interest

For the purpose of this subsection suppose that the set of clusters $C_{1:k}$ has been ordered, so that C_1 contains the lowest entropy, while C_k the highest. This concept will be analysed again, while looking at soft and hard detection ranges, in the next section. Moreover, recall that entropy is bounded (see (4.10) in Section).

Suppose that there is $t, 1{\leq}t{\leq}k$, such that connections inside clusters $C_{1:t}$ are very regular. As a result, if one calculates entropy of any cluster from $C_{1:t}$, it is close to 0. On the contrary, clusters from $C_{t+1:k}$ are populated with very irregular sequences; say their requests are distributed randomly on Ω. Subsequently, entropy of any $C_j, t{+}1{\leq}j{\leq}k$ is very close to $\ln(|\Omega|^2)$.

Moreover, assume there is an attacking sequence $s_a, 1{\leq}a{\leq}n_{VA}$, and there are two possible assignments:

1. sequence s_a has been assigned to any cluster from $C_{1:t}$
2. sequence s_a has been assigned to any cluster from $C_{t+1:k}$

Initially, consider (1) from above. Entropies inside $C_{1:t}$ are peaked at popular transitions, known to legitimate users only. Subsequently, transitions observed in s_a mostly "avoid" peaks inside $C_{1:t}$, which results in very low probability of transitions, and it is reasonably easy to detect s_a inside any of the cluster from $C_{1:t}$.

On the contrary, consider(2) from above. Observe that, making rare transitions for $C_{t+1:k}$ might have *very high* probability, comparing to other sequences in $C_{t+1:k}$. Therefore, it is important to minimise the effect of interest groups encoded in $C_{t+1:k}$, by connection reallocation.

In principle, consider two clusters $C_i \in C_{1:t}$, and $C_j \in C_{t+1:k}$. Moreover, suppose that C_j contains only one sequence (thus the same index) s_j. Subsequently, suppose that reallocation s_j from C_j to C_i decreases entropy of C_i by some factor. Furthermore, because $h_i < h_j$, then s_j improves h_i by inclusion of s_j into C_i. As a

result, C_j is being removed as it does not contain any sequence. This process causes the following (desired) effects:

1. Increases probability of popular transitions
2. Minimises (4.14)
3. Makes detection faster, by decreasing the number of clusters

Therefore, the main advantage of reallocation is further minimisation of (4.14). Moreover, attacking sequences will be assigned to clusters with peaks, rather than flat, and their transitions will have lower probability. Finally, detection of sequences is faster, as the number of clusters decreases with time.

4.7 Soft and Hard Detection Ranges

As it will be shown in the next Chapter, legitimate sequence tend to be allocated to low-entropy clusters. Moreover, the opposite holds for the attacking sequences, which find their allocation in higher-entropy clusters. Therefore, it is assumed that some of the detection techniques described in the subsequent sections should vary from cluster to cluster. Essentially, suppose that the set of clusters $C_{1:k}$ has been ordered, so that C_1 contains the lowest entropy, while C_k the highest. Clearly, because C_1 attracts many legitimate users, detection techniques should be relaxed there (presumably, should not be applied at all). On the contrary, C_k does not accommodate many legitimate users. Therefore, during the attack detection measurements should be strict (presumably, block each connection).

Therefore, given these consideration, classification region is introduced as two ranges: "hard" $H_H{=}\{h_1, \ldots, h_t\}, 1{\leq}t{<}k$ and "soft" $H_S{=}\{h_{t+1}, \ldots, h_k\}$, and $H_{1:k}{=}H_H \cup H_S$. For "soft" clusters less restrictive thresholds are applied, and the opposite holds for "hard" clusters; as described above. Recall that in Section 2.6, the application of ROC curve has been explained for IDS performance measurement. Moreover, it has been shown that performance of IDS depends on some varying parameters. Therefore, in this thesis this parameters will be t, which is the border between two ranges. Initially, t will be set to k, so that the entire $H_{1:k}$ is validated with the "soft" methods of detection. This will allow for acceptance of many legitimate users (i.e. very low FP rate). Conversely, this setting will lead to numerous instances of attacking sequence misclassification (i.e. relatively low TP). Subsequently, t will iteratively be moved towards lower-entropy clusters, until the entire set $H_{1:k}$ is validated with the "hard" detection setting. As a result, IDS will

notice the improvement of malicious sequences classification (i.e. relatively higher TP). However, the stricter detection implies misclassification of the legitimate users (i.e. relatively higher FP rate).

4.8 Allocation of Attacking and Validating Sequences

In this section, a method is described by application of which, the validating and attacking sequences are assigned into the set of clusters $C_{1:k}$. The steps described in this section apply to macro-interest and micro-interest methods explained in Sections 4.1.1 and 4.1.2, respectively.

Therefore, having clustered the connections form T, the connections from V and A should be assigned to the set of clusters $C_{1:k}$. Subsequently, one can be interested in the posterior probability of choosing a cluster $C_j, 1 \leq j \leq k$, given a set of transitions observed in the sequence s_i. Observe, that this can be expressed with Bayes formula as:

$$P(C_j|s_i) = \frac{P(C_j, s_i)}{P(s_i)} = \frac{P(s_i|C_j)P(C_j)}{P(s_i)} \tag{4.18}$$

For the composition of clusters $C_{1:k}$, $P_{1:k}(s_i)$ is a normalising constant, and is therefore omitted. As a result, (4.18) can be rewritten as:

$$\begin{gathered} P(C_j|s_i) \propto P(s_i|C_j)P(C_j) \\ \text{posterior} \propto \text{likelihood} \times \text{prior} \end{gathered} \tag{4.19}$$

Moreover, having assumed non-informative (uniform) prior distribution of selecting a cluster, the following holds:

$$\begin{gathered} P(C_j|s_i) \propto P(s_i|C_j) \\ \text{posterior} \propto \text{likelihood} \end{gathered} \tag{4.20}$$

Note that the choice of using uniform prior makes sense for assignment of sequences into the set of clusters $C_{1:k}$. Essentially, a weight is not imposed on the choice of clusters. It stems from the fact that sequences can join any cluster, and "popular" clusters should not be favoured. On the contrary, if the highly populated clusters were given more weight, attacking sequences could be mistakenly assigned to these clusters. Moreover, observe that this is a conventional approach in intrusion detection. For example, Lane et al. [43] or Zanero et al.[118] used non-informative prior for the problem of detection users, violating expected profile of computer com-

mands. While prior in [43] refers to prior beliefs on whether a user is legitimate or malicious, in [118] prior distribution is used (same as here) to denote probability of a particular class.

Subsequently, to find a cluster whose residing sequences resemble a sequence $s_i, 1{\leq}i{\leq}n_{VA}$ best, a popular technique based on maximum likelihood principle is introduced [43, 49]. In simple terms, the best model (i.e. cluster) is the one, under which an observation attains *the highest likelihood*. The likelihood of connection s_i inside $C_j, 1{\leq}j{\leq}k$ should be thought of as:

$$\mathcal{L}(s_i|C_j){=}P_j(s_i^1)\prod_{t=2}^{n_i}P_j(s_i^t|s_i^{t-1}) \tag{4.21}$$

Subsequently, the probability of generating sequence s_i is calculated for each cluster, and then C_j is chosen for which (4.21) attains the highest value:

$$C_j{=}\underset{C_1,C_2,...,C_k}{\arg\max}\ \log\mathcal{L}(s_i|C_{1:k}) \tag{4.22}$$

where $\mathcal{L}(s_i|C_{1:k})$ is the likelihood of observing connection s_i inside the set of clusters $C_{1:k}$.

4.9 Strategies of Attacking Hosts Against Macro-Interest Approach

Recall that, when macro-interest approach is considered, the focus is put on the hosts changing interest among web categories, rather than actual web objects. Moreover, suppose that having requested a link from category $c \in \Omega$, a programmed zombie faces a decision-making problem whether with probability p_R to *remain* inside the same category (i.e. request a link from c-th category again), or with probability $p_M{=}1{-}p_R$ *move* to any of the remaining categories (i.e. request a link from category $c_m \in \{\Omega\backslash c\}$). Given this decision-making problem, two attacking strategies are considered: rarely-changing and frequently-changing zombies.

4.9.1 Rarely-changing Hosts

Suppose that one wants to generate attacking sequences A similar to T. Because information encoded in $C_{1:k}$ remains unknown for attackers, suppose they have found out how many different categories are visited on average inside T. As a result, while analysing T, a vector e_T has been calculated containing the expected number of

visited categories v_c^r, given the number of requests $r \in \{r_\wedge, \cdots, r_\vee\}$, such that e_T=$[\mathbb{E}\{v_c^{r_\wedge}\}, \mathbb{E}\{v_c^{r_\wedge+1}\}, \ldots, \mathbb{E}\{v_c^{r_\vee}\}]$, where $\mathbb{E}\{v_c^{r_\wedge+t}\}$ denotes the expected number of visited categories after $r_\wedge + t$ requests. Subsequently, p_R=0.92, p_M=0.08 have been estimated to create randomly generated data set A_{C_R}, whose corresponding e_A is similar to e_T. Therefore, these zombies will *remain* for many more requests in one category, and are similar to sequences inside T.

4.9.2 Frequently-changing Hosts

Frequently-changing hosts are the ones that tend to change categories more frequently comparing to the rarely-changing zombies. Specifically, for this type of attacking hosts p_M=p_R=0.5. In other words, while making requests zombies will stay inside or move to another category with equal probability p_M=p_R=0.5. Observe that a frequently-changing host changes categories more frequently, than a rarely-changing host mentioned above, and thus its behaviour reminds human activity less. However, it is not as naive as a bot that could be generated with p_M=$\frac{1}{|\Omega|}$. Moreover, data set with their visits is denoted with A_{C_F}.

4.10 Strategies of Attacking Hosts Against Micro-Interest Approach

For many years, low likelihood of any activity has been considered anomalous, as observed in the pioneered work by Denning et al. [13]. As a result, A could be composed of sequences whose requests are uniformly distributed, and do not requests objects encoded in Ω. However, it is crucial to define realistically the attacking hosts' strategies. Therefore, we have decided to introduce additional attacking strategies that remind human behaviour as follows.

Set A_R is introduced containing sequences generated in a random walk fashion. Fundamentally, every time a request is to be made by a bot, a set of possible choices is presented from Ω (i.e. popular links) that have originally been recorded for that transitions. As a result, sequences in A_R avoid unobserved transitions. Moreover, this strategy has been chosen, as some of the technical solutions might miss this attack [25, 24] (see Section 3.1.4 for details).

In addition, there are two sets of sequences generated from *the most* popular object that have contributed the most to selection of centroids in Ω. These are two attacking strategies: naive and more reasonable, resembling human behaviour. For these two sets, sequences will uniformly request popular objects whose labels are

stored in Ω. First of all, reasonable attackers will know the most popular starting link depending on a number of requests they have decided to browse a website for. Therefore, these sequences will *not* wander across a website. Rather, they will request a small set of very popular objects in a random fashion. In addition, they will know how many *different* objects are allowed to use, so that they differ from the naive hosts. The set containing naive attackers is referred to as A_N, while A_H denotes human-like sequences. Moreover, recall that A_N and A_H have been motivated by results of [45], in which these strategies have been most successful (see Section 3.2 for details).

4.11 Detection of Attacking Hosts Against Macro-Interest Approach

It is assumed that internet attackers cannot have access to the profile generated by legitimate users. As a result, they cannot know how many times and in which order the correct categories have been requested. Therefore, it is crucial to develop a measure to check how "anomalous" one sequence behaves. Moreover, recall that there are two strategies of attacking hosts: hosts rarely-changing categories and stored inside A_{C_R} data set, and host frequently-changing categories and stored in A_{C_F} data set.

4.11.1 Mahalanobis Distance

To measure expected categorical interest among connections within a cluster, one can use Mahalanobis distance. It scales weights of each attribute, by inclusion of inverse of variance. This implies that attributes with low variability receive more weight comparing to components with high variability. Moreover, correlation between categorical visits is taken into consideration as well.

As a result, for each cluster $C_i, 1{\leq}i{\leq}k$ a corresponding covariance matrix Σ_i is calculated, together with a vector of average categorical requests $\boldsymbol{\mu_i}{=}[\mu_i^1, \mu_i^2, \ldots, \mu_i^{n_C}]$. Subsequently, each *training* connection $s_j, 1{\leq}j{\leq}|C_i|$ from C_i, is transformed into a vector form $v_j{=}[v_j^1, v_j^2, \ldots, v_j^l, \ldots, v_j^{n_C}]$, where each v_j^l denotes how many times l-th category has been requested during session s_j. As a result, *training* Mahalanobis distance can be calculated in the following way:

$$d_M(v_j, \boldsymbol{\mu_i}){=}\sqrt{(v_j - \boldsymbol{\mu_i})^T \Sigma_i^{-1} (v_j - \boldsymbol{\mu_i})} \tag{4.23}$$

Observe that when $\Sigma_i = \mathbb{I}$ ($\Sigma_i \equiv \text{diag}(\sigma_i^1, \ldots, \sigma_i^{n_C})$), then the above is just the Euclidean (weighted Euclidean) distance between the number of categories visited during a validating sequence s_j, and the expected training profile $\boldsymbol{\mu_i}$. Subsequently, vectors of Mahalanobis distances m_i=$[m_i^{r_\wedge}, m_i^{r_{\wedge+1}}, \ldots, m_i^{r_{\wedge+t}}, \ldots, m_i^{r_\vee}]$ are obtained, where each t-th component contains the maximum Mahalanobis value observed inside C_i for t-request-long sequences.

Therefore, every time a sequence $s_j, 1 \leq j \leq n_{VA}$ has been assigned to cluster C_i its Mahalanobis distance $d_M(s_j, C_i)$ will be computed. Recall that n_j denotes number of requests inside s_j. As a result, if $d_M(s_j, C_i) \leq m_i^{n_j}$, then s_j will be marked as legitimate connections. Otherwise, s_j will be marked as anomalous.

Observe that sequences inside A_{C_R} are similar to legitimate sequences, which "remain" inside popular category l_t (i.e. keep on requesting objects from l_t). This implies that the number of l_t requests, has a *larger* variance comparing to the unpopular categories. Specifically, observe that large variance is caused by short and long sequences, whose total number of l_t requests changes with the number of requests observed during training. Moreover, it is assumed that interest in web categories can be explained with covariance matrix in (4.23). In other words, legitimate users visit frequently popular categories in a cluster, and refrain visiting others, in a similar pattern. As a result, detection based on Mahalanobis distance is more effective against the frequently-changing hosts stored inside A_{C_F}, which fail to repeat legitimate behaviour measured with (4.23).

4.11.2 Mutual Information and Statistical Independence as Anomalous Measurement

Mutual information between two discrete random variables a, b is defined in the following way:

$$I(a;b) = \sum_{a,b \in \Omega} \sum P(a,b) \log \left(\frac{P(a,b)}{P(a)P(b)} \right) \tag{4.24}$$

Mutual information provides a measure of independence between two random variables. It is equal to zero (i.e. $I(a;b)$=0) for independent variables, and increases when knowing realisation of one variable reduces uncertainty of the other. As a result, a measure is introduced based on statistical independence.

For each cluster C_i a two-dimensional mutual information vector i_i is introduced, where each j-th column contains $i_i^{1,j}$ minimum and maximum $i_i^{2,j}$ value of mutual information while requesting j-th category.

As a result, while analysing sequences from T, mutual information is calculated between pairs of two consecutive categorical requests. In other words, for

a sequence s_l=$(s_l^1, s_l^2, \ldots, s_l^t, \ldots, s_l^{n_l}), 1 \leq l \leq n_T$, dependencies are calculated in the following way: $I(s_l^{t-1}; s_l^t)$.

Because it is assumed that human users will follow similar patterns of behaviour in V as in T, then it should not be the case that two different categorical requests from sequence s_l, say s_l^{t-1}, s_l^t, produce value of mutual information greater or smaller than the corresponding i_i^{2,s_l^t} and i_i^{1,s_l^t}. Otherwise, it is assumed that transition s_l^{t-1}, s_l^t has never been observed in C_i, and is anomalous.

Recall that it has been reported that the same-page attack is detected, and prevented easily with a software-based solution (see Section 3.1.4). Therefore, sequences inside A_{C_R} and A_{C_F} *do* change categories to avoid being detected by the software solution. However, it is assumed that mutual information of transitions between categories should fall outside the range of i_i. Therefore, this detection measure should perform well against A_{C_R} and A_{C_F}.

4.11.3 Likelihood of the Same-category Segment

Suppose there is a sequence s_i=$(s_i^1, s_i^2, \ldots, s_i^l, \ldots, s_i^{n_i}), 1 \leq i \leq n_T$ assigned to a cluster $C_j, 1 \leq j \leq k$. For an arbitrary $t \neq 0, r_\wedge \leq l < t \leq n_i$, a segment $s_i^{l:t} = \{s_i^l, s_i^{l+1}, \ldots, s_i^t\}$ is introduced, denoting the longest substring observed inside s_i after t-th request, and composed of the same elements s_i^l, such that $s_i^l = s_i^{l+1} = \ldots = s_i^t$. As a result, one could be interested in the posterior probability of observing *the length* of segment $s_i^{l:t}$ among other connections from C_j, which can be expressed with the following formula:

$$\mathbb{P}_j(t-l|t) = \frac{\mathbb{P}_j(t, t-l)}{\mathbb{P}_j(t)} \tag{4.25}$$

In (4.25) $\mathbb{P}_j()$ is used instead of $P_j()$ to make distinction between distributions of requests in clusters. Subsequently, $\mathbb{P}(l_S, n_r)$ denotes a joint probability of observing a *length* of a segment consisting of the same elements l_S, after some number of requests n_r. Moreover, the sample space for $\mathbb{P}(l_S, n_r)$ must be defined as

$$\Omega_S = \{(l_S, n_r) \in A; A = \{(r_\wedge, r_\wedge), (r_\wedge, r_{\wedge+1}) \ldots, (r_\vee, r_{\vee-1}), (r_\vee, r_\vee)\} \tag{4.26}$$

Clearly, given (4.26), it is apparent that different segments $s_i^{l:t}$ cannot be longer than the maximum number of requests $r_\wedge$.

Subsequently, for each cluster C_j, set L_j=$\{\log(\mathbb{P}_j(l_S|r_\wedge)), \log(\mathbb{P}_j(l_S|r_{\wedge+1})), \ldots, \log(\mathbb{P}_j(l_S|r_\vee))\}$ is introduced, containing minimum logarithmic values of (4.25), given number of requests. As a result, once a sequence has been analysed, the logarithm of its longest same-element segments posterior probability is checked against

L_j. If it is smaller than values in L_i, then the sequence is marked illegitimate.

Moreover, observe that this technique performs very well against sequences inside A_{C_R}, which prefer to frequently request objects from a category. Sequences inside A_{C_R} might guess which category is popular, but the number of the same-category request remain to attackers unknown.

4.12 Detection of Attacking Hosts Against Micro-Interest Approach

In this section, two approaches have been explained against the attacking sequences. Recall, that in Section 3.2 it has been assumed that measurement of anomaly should be: (i) attack-strategy independent (i.e. should be able to detect different browsing strategies chosen by attackers), (ii) and attack-scale independent (i.e. regardless of the number of browsing bots, the attempt should be discovered, especially when this number is very small). Initially, a metric based on Bayes Factor (BF) is presented that is indeed (ii), but can under perform when it comes to (i). Subsequently, a measurement based on the likelihood analysis that is both (i) and (ii) is described.

4.12.1 Detection Based on Bayes Factors

Having assigned sequences from V, A to $C_{1:k}$, there are two hypotheses about a sequence's intention: $\mathbb{L}$ legitimate or $\mathbb{A}$ attacking. The odds of one hypothesis against the other given observation O can be calculated in the following way with BF [40]:

$$\frac{P(H_1|O)}{P(H_2|O)} = \frac{P(O|H_1)P(H_1)}{P(O|H_2)P(H_2)}. \tag{4.27}$$

Therefore, suppose that a sequence $s_i, 1{\leq}i{\leq}n_A$ has been assigned to a cluster $C_j, 1{\leq}j{\leq}k$, and one wants to measure whether connection s_i is attacking or legitimate. Recall that $P_j(a,b)$ denotes the training joint distribution for cluster C_j. Similarly, an empirical joint distribution $\dot{P}_j(a,b)$ is obtained by analysing arriving sequences from V, A assigned to C_j. In addition, *prior* distributions $\pi_j(C_j)$, $\dot{\pi}_j(C_j)$ are introduced, such that $\pi_j(C_j) = \frac{f_j}{f_j+\dot{f}_j}$ and $\dot{\pi}_j(C_j) = \frac{\dot{f}_j}{f_j+\dot{f}_j}$, where f_j,$\dot{f}_j$ denote relative frequency of sequences in clusters corresponding to $P_j(a,b)$, and $\dot{P}_j(a,b)$ respectively. As a result, classification of legitimate intention is equivalent to measuring whether the patterns of transitions inside $s_i \in V \cup A$ are more similar to $P_j(a,b)$ than to $\dot{P}_j(a,b)$. As a result, because s_i represents observation, then BF is

calculated in the following way:

$$\underset{\mathbb{A}}{\Lambda_{\mathbb{L}}} = \frac{P_j(s_i|C_j)}{\bar{P}_j(s_i|C_j)} \tag{4.28}$$

It has been shown [51] that when $1 \leq \log_{10}(\underset{\mathbb{A}}{\Lambda_{\mathbb{L}}}) < 3$ then evidence is barely worth mentioning, and when $3 \leq \log_{10}(\underset{\mathbb{A}}{\Lambda_{\mathbb{L}}}) < 5$ it is substantial, in favour of *legitimate* intention. Values below 1 will introduce inverted interpretation supporting attacking intention [51]. Therefore, two thresholds are chosen $\lambda_{\mathbb{S}}$=-3 and $\lambda_{\mathbb{H}}$=-1. Thus, to mark connection as *attacking*, $\log_{10}(\underset{\mathbb{A}}{\Lambda_{\mathbb{L}}}) < \lambda_{\mathbb{S}}$ and $\log_{10}(\underset{\mathbb{A}}{\Lambda_{\mathbb{L}}}) < \lambda_{\mathbb{H}}$ will have to be met for "soft" and "hard" clusters, respectively.

4.12.2 Detection Based on Likelihood Analysis

Having assigned validating and attacking sequences to $C_{1:k}$, there are two hypotheses about a sequence's intention: legitimate or attacking. Therefore, a training log-likelihood matrix $\ell_{i,j}$ is obtained, such that $1 \leq i \leq k$ and $r_\wedge \leq j \leq r_\vee$. In other words, $\ell_{i,j}$ contains log-likelihood values for each cluster (i.e. $1 \leq i \leq k$) and between $r_\wedge$ and $r_\vee$ requests. Each time a host requests an object at a website, her entire session is assigned to a cluster C_i given (4.22), and likelihood of the sequence calculated. If it is below the value stored in $\ell_{i,j}$, the connection is marked anomalous.

4.13 Summary

In this chapter a detailed description has been given of methods applied to the problem of unsolicited browsing. Initially, it has been shown that web interest can be divided into two categories: (i) macro-interest and (ii) micro-interest. Specifically, (i) denotes interest into logical categories of a website, while (ii) denotes interest in URI's of web objects inside the categories. Moreover, it has been shown that for micro-interest a data processing step is required to perform dimensionality reduction. Detailed description has been provided in Table 4.1.

Subsequently, an entropy minimising clustering approach and two ways of calculating the number of clusters have been demonstrated. Afterwards, strategies of the attacking hosts have been presented against the legitimate profiles of actual users. Moreover, statistical measures have been introduced that could detect connection intention.

In the next chapter, the results obtained by the application of the methods are presented. Most importantly, it is shown that distribution of sequences across clusters allows for detection of ongoing attack. Moreover, it is shown that

Feature	Macro-Interest Approach	Macro-Interest Approach
Web Interest	monitoring of requests at logical web categories level *only*, e.g. how often a user requested web objects inside weather category, how often an image or pdf file was downloaded, etc.	monitoring of popular unique URI's of existing web objects
Dimensionality Reduction	focus put at categorical level (instead of all existing URI's) provides significant reduction of space	only popular links are observed, and unpopular represented by a dummy link (see details in Section 4.1.2)
Advantages	does not require initial stage of popular link selection, categorical set is fixed and easily expendable	introduces another level of difficulty as transitions among web links are unknown, let alone among popular links that are difficult to discover; performs relatively well against different attacking strategies (see next chapter for details)
Disadvantages	as it transpires it the next chapter, focus at logical categories is insufficient against sophisticated attacking strategies	resource intensive, requires more calculations

Table 4.1: Comparison of Micro and Macro Approach

the methods presented for (i) cannot be applied for narrow websites (i.e. containing a couple of categories, with multiple links). However, they provide promising results for websites with multiple categories. Finally, it has been presented that analysis of likelihood of each connection, is a measure that regardless of attacking strategies and scope, provides reasonable good results of intention classification.

Chapter 5

Results

In this chapter, the results of the clustering algorithm and implementation of detection methods are demonstrated. Recall that there are two approaches to data analysis: (i) macro-interest and (ii) micro-interest. Firstly, (i) is considered and initially, sequence distribution across the clusters is presented. It is shown that distribution of legitimate users is very regular, and attacking sequences differ from training distribution. Subsequently, application of statistical measurements it presented. Afterwards, (ii) approach is analysed. Initially, stability of the clustering algorithm is presented. It is shown that for different permutations of training sequences, very similar clusters are obtained. Next, it is shown that, again, distribution of sequences across clusters allows for detection of on-going attack. Recall that there are two methods implemented for the micro-interest approach, based on Bayes Factors (BF) and likelihood analysis. Initially, the results of BF application are presented. It is shown that the method performs well against varying browsing strategies chosen by attackers. Subsequently, results of likelihood analysis are described. It is shown that regardless of strategies and the number of attacking hosts, the method is able to detect harmful intention. This chapter concludes with analysis of the obtained results.

5.1 Sequence Distribution for Macro-Interest Approach

As it has been mentioned, it is assumed that if there is a set of new links or files uploaded on-line that propel, say, "behaviour" $C_i, 1 \leq i \leq k$, then one could expect a surge of connections for C_i within a limited time period. Similar assumptions and observations have been reported in scientific literature [35, 101]. Therefore, it is expected that regularity of legitimate users is perceived as interests in the same web

objects, and the same way these are requested.

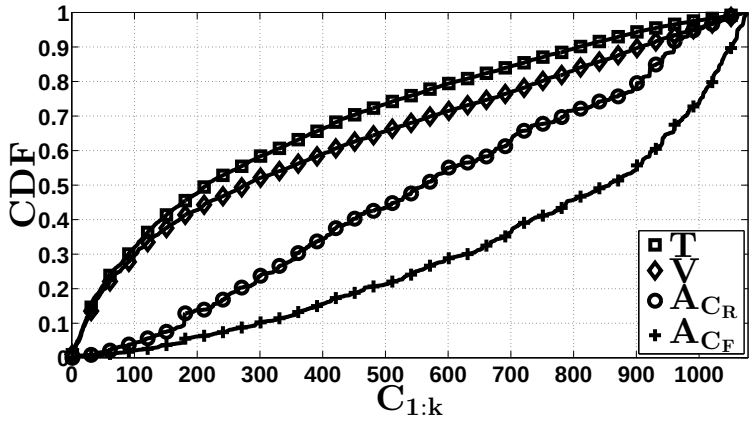

Figure 5.1: CDF of sequence assignment in MSNBC trace. Observe that training and validating distribution of legitimate users are very similar, and approach the maximum faster, comparing to attacking sequences

Similarly, the same regularity has been observed for MSNBC data set. Specifically, suppose that $C_{1:k}$ has been ordered as before (see Section 4.7 for details), so that C_1 contains the lowest entropy, while C_k the highest. Afterwards, one could be interested in an event describing "how fast" a sequence is assigned to the set of clusters $C_{1:k}$. Furthermore, one could expect legitimate sequences to be assigned "faster", comparing to attacking sequences residing in high-entropy clusters.

For this purpose, a cumulative distribution function has been calculated and plotted in Fig. 5.1. From Fig. 5.1 one can learn that training and validating sequence occupy low-entropy clusters. Moreover, distributions of training and validating sequences are similar. On the contrary, attacking hosts cannot repeat popular transitions, and end up being allocated into high-entropy clusters. Observe that this process allows for detection of the incoming attack. Clearly, the currently present attacking hosts deviate from the expected sequence distribution indicating malicious activity.

Recall that attacking strategies have been defined to mimic actual browsing patterns. Therefore, the attacking hosts tend to find themselves in the low-entropy clusters as well. On the other hand, a fraction of legitimate sequences tend to populate high-entropy clusters. As a result, there is no cluster $C_t, 1 \leq t < k$ for which majority of legitimate sequences populate $C_{1:t}$, and most of the attacking hosts have been assigned into $C_{t+1:k}$. As a result, it is impossible to classify all sequences visiting $C_{1:t}$ as legitimate, and attacking otherwise (i.e. allocated to $C_{t+1:k}$); to obtain reasonable classification results. Therefore, the statistical measurement of intention

classification have been introduced, and the results of their application presented in the next section.

5.2 Intention Classification for Macro-Interest Approach

Recall that there are three statistical measurements introduced in the previous chapter. These are based on: (i) Mahalanobis distance (see Section 4.11.1), (ii) mutual information (see Section 4.11.2), and (iii) likelihood of the longest same-category segment (see Section 4.11.3). Moreover, recall that there are two strategies of attacking hosts: hosts rarely-changing categories and stored inside A_{C_R} data set, and host frequently-changing categories and stored in A_{C_F} data set. Moreover, recall that it has been observed in Section 4.11 that (i) and (ii) should detect well sequences inside A_{C_F}. In addition, sequences inside A_{C_R} should be detected with (ii) and (iii). The results of statistical measurements application are presented below. Moreover, recall that there is a division for "hard" and "soft" clusters (see Section 4.7 for details). In general, in "soft" clusters, (ii) measure is used, and is supported by (i) and (iii) in "hard" clusters against sequences from A_{C_F} and A_{C_R}, respectively.

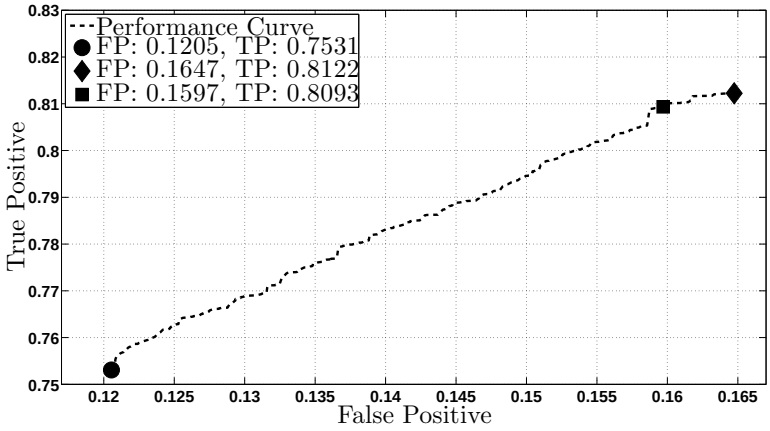

Figure 5.2: Performance curve of the detection algorithm against validating and frequently-changing hosts.

Figure 5.2 depicts the application of statistical measurements against sequences inside A_{C_R} and V. One can observe that the more clusters are checked against the supplementary measure, the more legitimate sequences are misclassified. At the same time, the number of attacking sequences correctly classified increases. Specifically, initially t has been set to the smallest entropy value, and the entire cluster set $C_{1:k}$ (except for the first cluster) has been checked with "soft" detection

technique. In this setting 88% of sequences from V, and 75% of sequences from A_{C_R} have been correctly classified (marked with a circle in Fig. 5.2). Subsequently, t has been iteratively moved to k, when the whole set of clusters has been checked against "hard" technique. In this setting, 84% of sequences from V, and 81% of sequences from A_{C_R} have been recognised (denoted with a diamond in Fig. 5.2). Moreover, the best setting of the algorithm has correctly classified 84% of sequences from from V, and 81% of sequences from from A_{C_R} (marked with a square in Fig. 5.2). Note that the performance of the algorithm in Fig. 5.2 shows results for possible values of $t \in \{1, \ldots, k\}$.

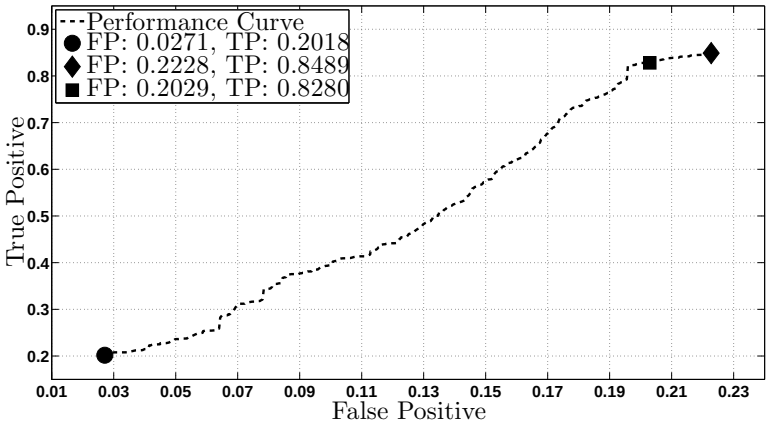

Figure 5.3: Performance curve of the detection algorithm against validating and rarely-changing hosts.

The performance of the algorithm changes against rarely-changing hosts, and has been depicted in Fig. 5.3. Initially, for $t = 1$, only around 3% of legitimate users are misclassified. However, at the same time, only 20% of attacking sequences are spotted (denoted with a circle in Fig. 5.3). Note that this is expected, as the attacking sequences inside A_{C_R} are very similar to V (see Section 4.9.1 for details). Once $t = k$, and the entire cluster set $C_{1:k}$ has been checked against "hard" technique, 83% of sequences from A_{C_R}, and 78% of sequences from V have been correctly classified (marked with a diamond in Fig. 5.3). The best performance achieves recognition rate of legitimate sequences at 80% and 83% of legitimate and attacking sequences, respectively; (denoted with a square in Fig. 5.2)

5.2.1 Result Analysis

Although the results depicted with ROC curves in Fig. 5.2 and Fig. 5.3 are promising, there are several issues with their applicability, however. Observe that when rate-limiting or dropping of attacking sequences is not prioritised (denoted with the

circles in Fig. 5.2, and Fig. 5.3), FP rate against A_{C_R} is satisfactory (i.e. around 3% of legitimate users are misclassified). However, the same does not hold for detection methods against A_{C_F}, in which 12% of legitimate sequences are misclassified. Furthermore, observe that the best performance of both approaches results in around 20% of misclassified legitimate and attacking sequences.

Thus, a novel approach should be implemented in order to improve performance. Recall that it has been mentioned that in general there are two ways of improving detection performance of IDS's [46]: by extension of feature space or improved data set decomposition. As a result, the micro-interest has been defined, as a way of feature space extension. The results of detection techniques for the micro-interest approach is described in the following section.

5.3 Clustering Results for Micro-Interest Approach

In this section the application of clustering algorithm and intention classification methods are described, for micro-interest approach. Initially, a stability of clustering approach has been described. It is shown that the algorithm arrives at similar results, in spite of using different permutations of sequences for training. Subsequently, sequence distribution in the composition of clusters is presented. Similarly to the macro-interest approach, the distributions of training and validating legitimate sequences are very regular and similar, unlike the attacking sequences. In addition, the result of sequence reallocation is provided. Specifically, it is shown that the process of reallocation allows for decreasing the number of clusters, which result in faster detection.

5.3.1 Stability of Clustering Algorithm

In order to validate whether the algorithm provides similar clustering results, 10 different clustering results $C_{1:k}^{1:j}, 1 \leq j \leq 10$ have been obtained and containing different permutations of sequences for each of the analysed data set. Subsequently, one could be interested to measure differences between probability distributions P_i^j and Q_i^j obtained from different batches, for the same cluster C_i. To provide result of distances among distributions, a cumulative distribution function $F_J(JSD)$ has been obtained expressing probability of different values of $JSD(P_i^j||Q_i^j)$ given different pairs of distributions P_i^j and Q_i^j. From Fig. 5.4 one can learn that most of the differences between distributions will be below 0.143, which denotes very similar clusters. Therefore, the empirically obtained results in Fig. 5.4 show that different permutations of sequences should provide similar clustering results.

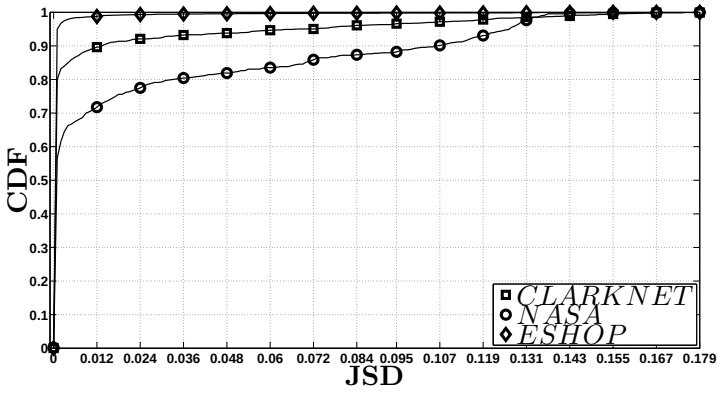

Figure 5.4: Performance curve of the detection algorithm against validating and rarely-changing hosts.

5.3.2 Sequence Distribution for Micro-Interest Approach

Recall that once the sequences have been clustered into the set of clusters, there are many interest groups that are not frequently populated. As a result, the reallocation of sequences is introduced based on entropy minimisation and likelihood maximisation (see Section 4.6.4 for details). One of the advantages of the reallocation process is the decreased number of clusters, which provides the same detection capabilities.

Essentially, the initial sequence distributions have been presented in Fig. 5.5 - Fig. 5.7. Again, given the fact that attacking strategies are very similar to legitimate behaviour (see Section 4.10 for details), their distribution of sequences across clusters is close to the legitimate profile. However, it still differs from the validating distribution of legitimate sequence that resembles the training distribution closely.

Subsequently, a distribution of sequences has been presented for reallocated sequences in Fig. 5.8- Fig. 5.10. Process of reallocation decreases k (i.e. reduces number of unpopular clusters by reallocation their sequences into more skewer clusters), and is stopped when there is no further change in k. Clearly, if all sequences from a cluster have been removed to other clusters, then the cluster has to be removed from the collection $C_{1:k}$. As a result, the smaller space of clusters is obtained and detection of intention is faster. For each data set the results of reallocation algorithm have been presented in Fig. 5.8- Fig. 5.10.

Observe that again there is no cluster C_t, that allows for separation of sequences into legitimate and attacking. Therefore, again, it is impossible to classify all sequences visiting $C_{1:t}$ as legitimate, and attacking otherwise (i.e. allocated to $C_{t+1:k}$).

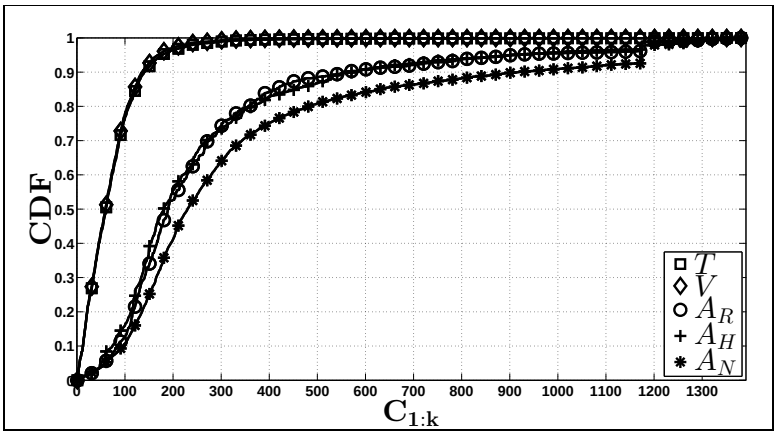

Figure 5.5: CDF of sequences in CLARKNET before reallocation

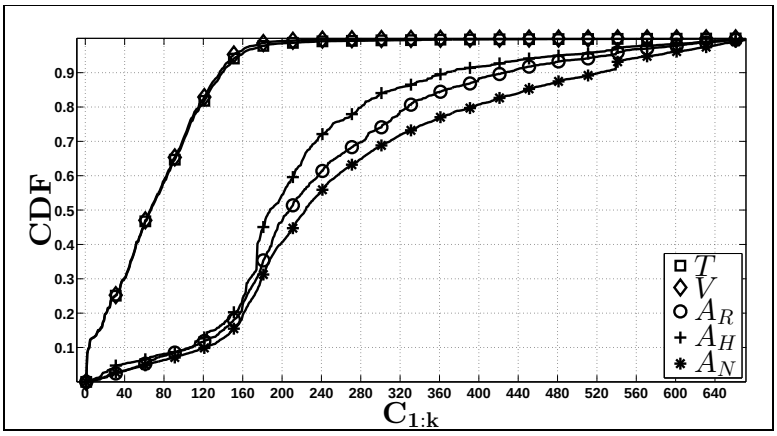

Figure 5.6: CDF of sequences in NASA data set before reallocation

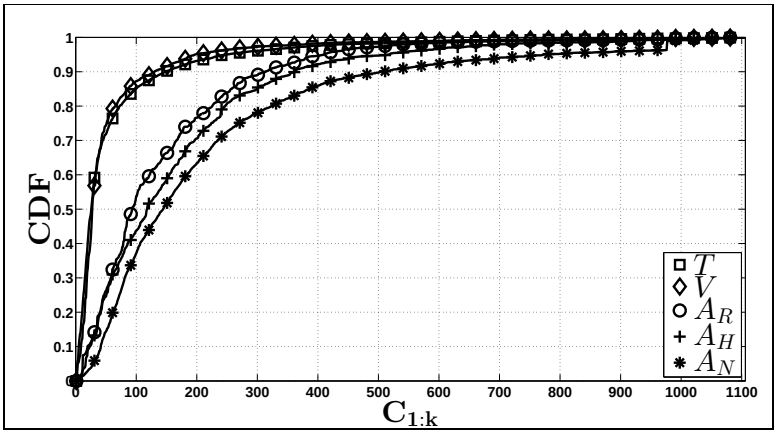

Figure 5.7: CDF of sequences in ESHOP data set before reallocation

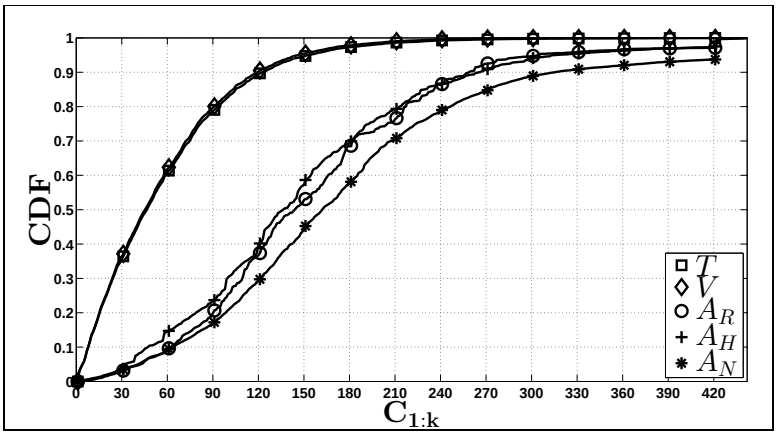

Figure 5.8: CDF of sequences in CLARKNET after reallocation

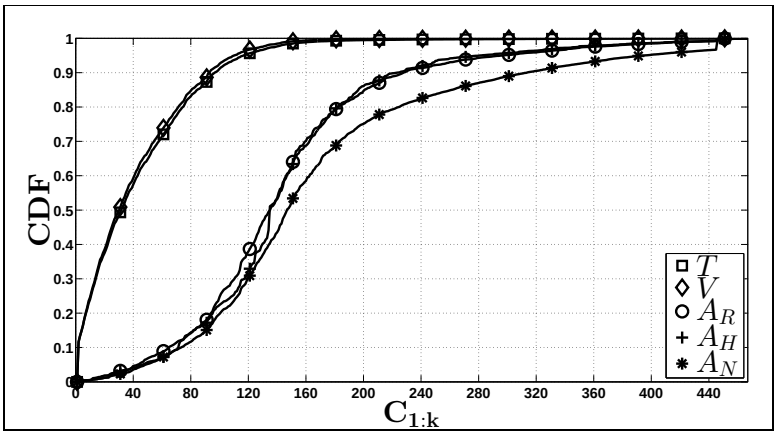

Figure 5.9: CDF of sequences in NASA data set after reallocation

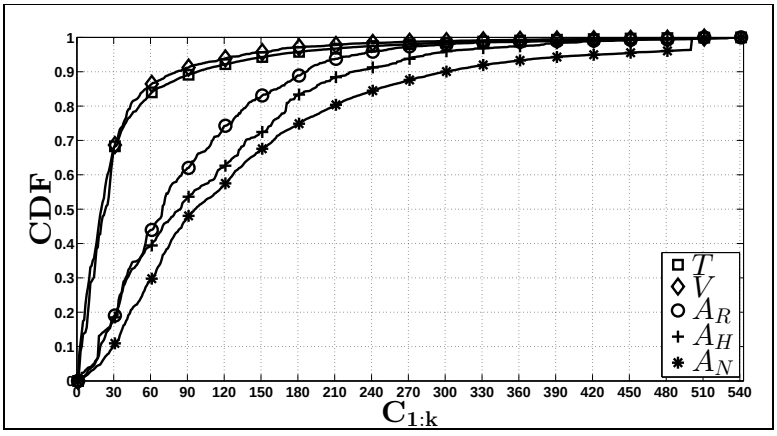

Figure 5.10: CDF of sequences in ESHOP data set after reallocation

5.4 Intention Classification for Micro-Interest

In this section, intention classification is described based on two approaches: (i) BF (see Section 4.12.1 for details) and (ii) likelihood analysis (see Section 4.12.2 for details). Description of the results begins with (i), followed by presentation of results obtained by (ii).

It is shown that approach (i) allows for detection of varying strategies of web browsing implemented by zombies. However, it cannot detect activity of lone unwanted crawlers. Therefore, methods using (ii) have been introduced in the previous section that allow for detection of scope-independent attack. In other words, regardless of the number of attacking hosts, the method allows for reasonably good classification.

5.4.1 Intention Classification with Bayes Factors

ROC curves in Fig. 5.11-5.13 have been obtained by iteratively moving t, as described in Section 4.7. Initially, each cluster has been validated with "soft" setting. This setting produces higher number of legitimate users (i.e. low FP), and implies a higher error of misclassification of attacking hosts (i.e. relatively low number of TP). Subsequently, t has been moved to higher entropy clusters, when entire $C_{1:k}$ has been validated with "hard" threshold, when the number of recognised malicious host increases. Note that validation with "hard" settings produces a high number of FP for ESHOP that contains relatively small number of training data (see Fig. 5.13). This phenomenon has been investigated further, and it is argued that the high number of FP stems from insufficient amount of data for training purposes. Specifically, recall that access to ESHOP data set has been obtained in order to validate the applied methods for a relatively smaller sample. Therefore, as it is shown in the subsequent section, higher FP rate stems from insufficient number of training sequences inside the data set.

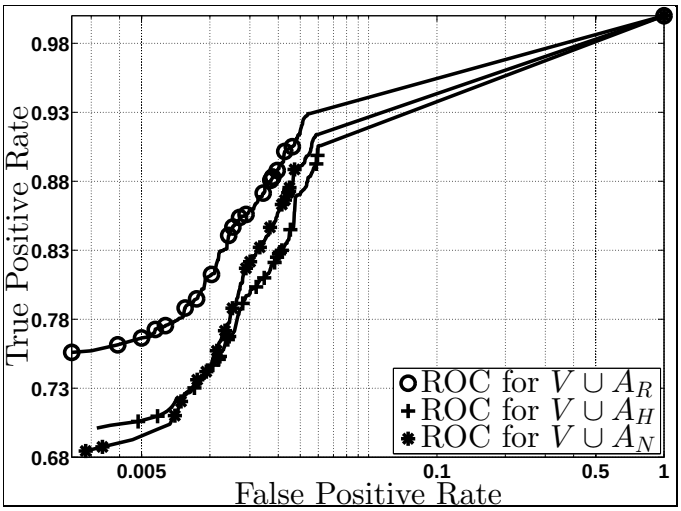

Figure 5.11: Performance curve for CLARKNET data set

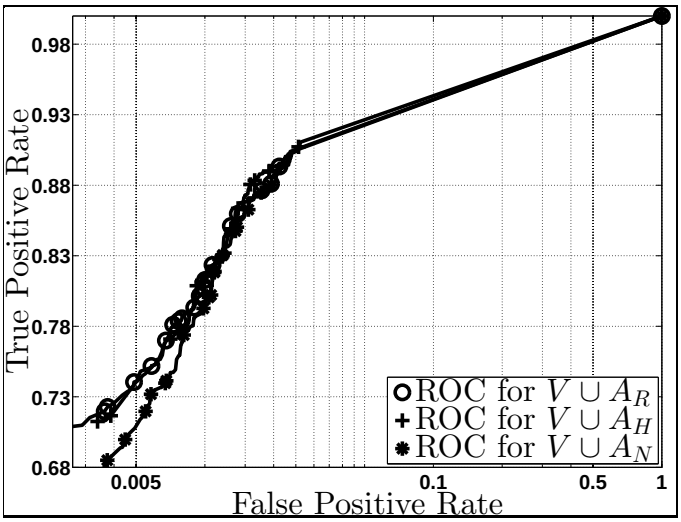

Figure 5.12: Performance curve for NASA data set

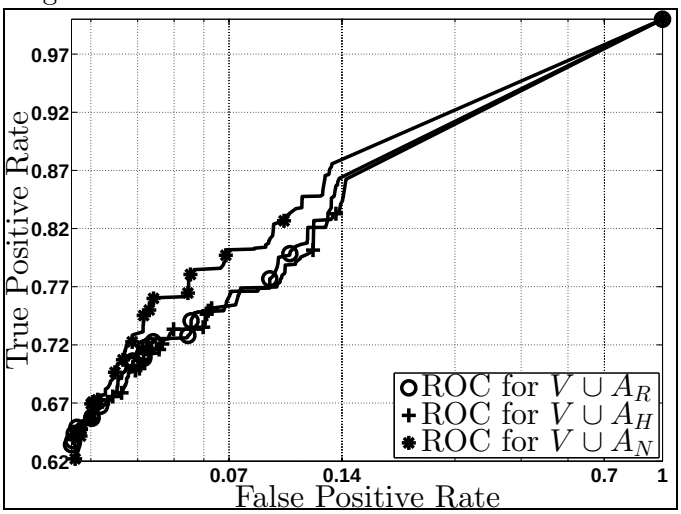

Figure 5.13: Performance curve for ESHOP data set

5.4.2 Length-Dependent Detection Analysis

To account for slightly higher FP rate for ESHOP data set, length-depended acceptance rate has been analysed. Clearly, it is essential to discover how many requests an attacking session is allowed to last for, before is classified as illegitimate. This result has been presented in Fig. 5.14-5.16. The three data sets presented in Fig. 5.14-5.16 have been validated with threshold $\lambda_{\mathbb{S}}$ only (see Section 4.12.1 for details), to minimise TP (i.e legitimate users misclassification rate). From Fig. 5.14-5.16 one can learn that with the increased number of requests, the number of attacking sequences that are let into the system drops. In other words, the more requests attacking hosts make, the better the system learns how to recognise them, while keeping high percentage of legitimate users in the system.

Interestingly, for the system with decoy links [62], 24 requests are required to detect 92% of attacking sequences, and 48 requests are needed for detection of all malicious hosts. As it stems from in Fig. 5.14 and Fig. 5.15, most attacking bots are detected after 25^{th} request, even though partial access to web log has been revealed (see Section 4.10 for details). Similarly, for ESHOP (see Fig. 5.16), after 18^{th} requests most bots are detected. In addition, as it has been observed before, for low-rate attacks against application servers, the think-time (i.e. time needed for learning of illegitimate behaviour) increases proportionally to the rate of the attack (i.e. the lower the rate, the longer it takes to an IDS to recognise attacking behaviour) [72]. The attacks presented in this work are most similar to request flooding described in [72], for which 30 requests are needed to discover legitimate nature of a host.

In addition, as it has been mentioned in Section 3.2, detector should work well regardless of the number of attacking hosts. Clearly, the more attacking hosts, the easier it is to learn their behaviour. Therefore, length-depended acceptance rate has been obtained for smaller attacking sample in case when n_A=$0.1n_V$ (i.e. the number of attacking hosts is smaller than legitimate). This is essentially important because it is hard to discover such a behaviour that can be generated by crawlers or e-mail address extractors. The results of this experiment have been presented in in Fig. 5.17-5.19. Again, most of the sequences are correctly classified.

Note sequences encoded in A_H give the attackers the best performance. Observe that this is the case for different data set, and for different scopes. Moreover, while analysing sequences distributions across the set of clusters for ESHOP (see Fig. 5.10 and Fig. 5.7) one can notice that distribution of attacking sequences is very close to the legitimate ones. This also has a negative effect on classification, and most of the short sequences generated for ESHOP data set have been misclassified.

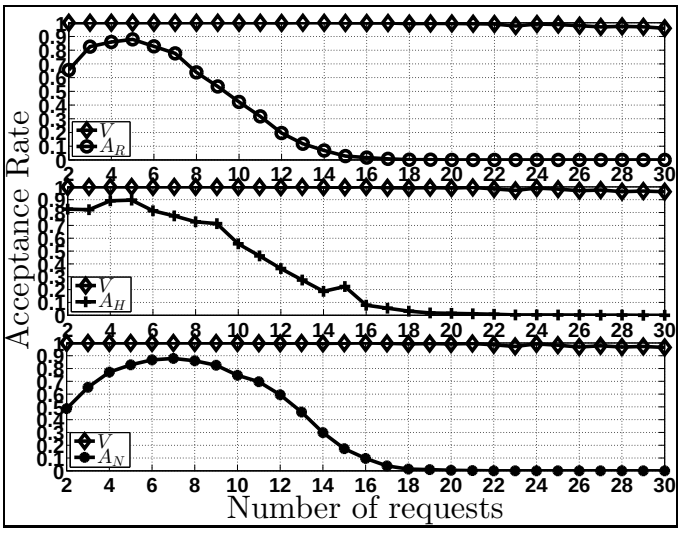

Figure 5.14: Length-dependent detection for CLARKNET data set

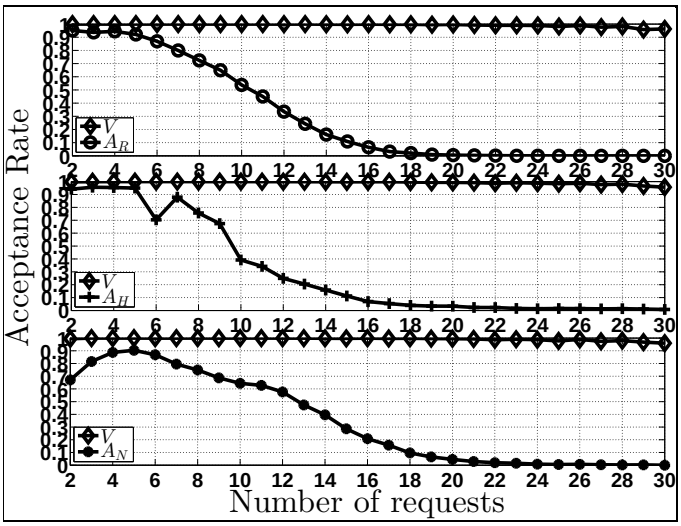

Figure 5.15: Length-dependent detection for NASA data set

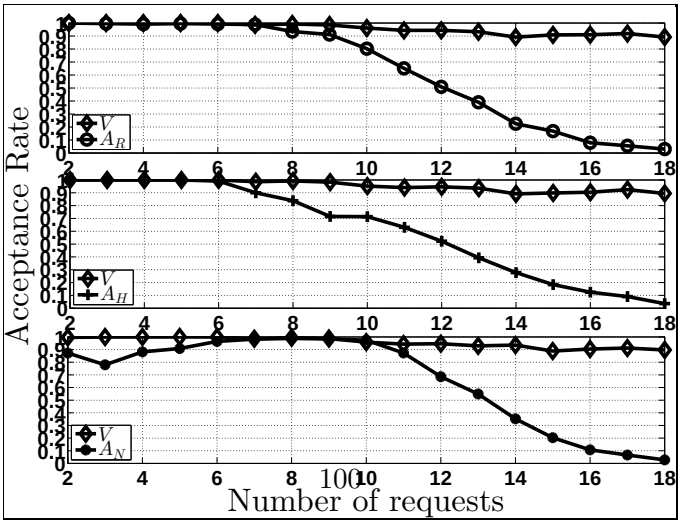

Figure 5.16: Length-dependent detection for ESHOP data set

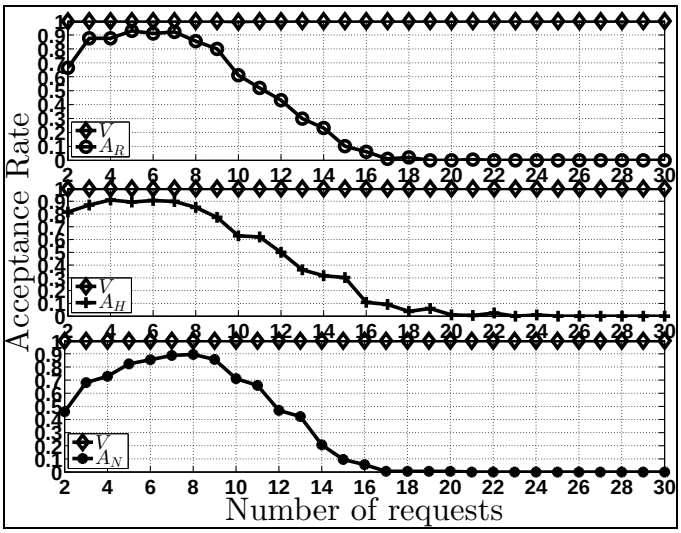

Figure 5.17: Length-dependent detection for CLARKNET data set, and smaller number of attacking hosts

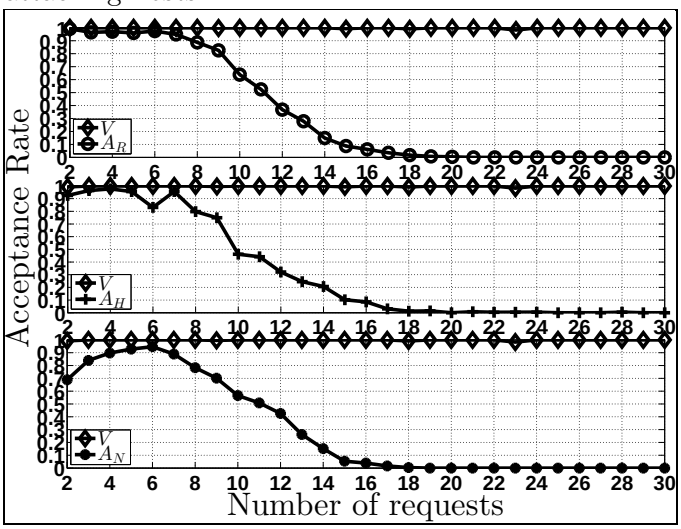

Figure 5.18: Length-dependent detection for NASA data set, and smaller number of attacking hosts

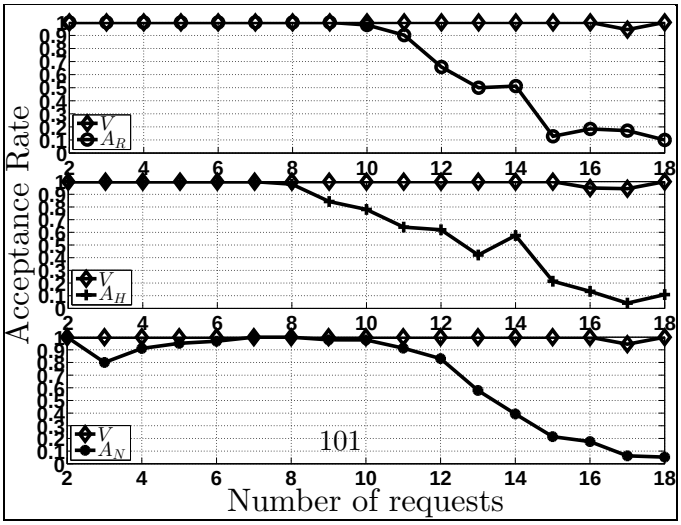

Figure 5.19: Length-dependent detection for ESHOP data set, and smaller number of attacking hosts

5.4.3 Intention Classification with Likelihood Analysis

In the previous section detection, methods utilising BF have been described. Observe that "knowledge" is required on strategies chosen by attackers. Therefore, once IDS has discovered that there is an ongoing attack, it classifies connection intentions based on patterns of actual traffic. Furthermore, it has been shown that measurement described in the previous section works well (even when attack traffic volume is not large). Therefore, measurement based on Bayes Factors are strategy-independent (i.e. achieves reasonably well performance for different attacking strategies).

However, observe that if there is a *single* crawler browsing through a website, its activity will not be spotted. Essentially, distribution of currently present sequences in the cluster set $C_{1:k}$ is still very similar to the training distribution. Furthermore, there is not much deviation in the incoming traffic (as it is caused by a single bot). Also, recall that detection measure should be strategy and scale independent, as it has been pointed out in section 1.

Therefore, the method based on likelihood analysis has been introduced in the previous chapter, and its application is described in this section. Specifically, the acceptance rate for attacking and legitimate sequences before the reallocation has been presented in Fig. 5.20 - Fig. 5.22. Having reallocated the sequences, an improvement in detection has been achieved, and shown in Fig. 5.23 - Fig. 5.25. This effect is especially significant for ESHOP data set, where improvement in detection is considerate (see Fig. 5.25).

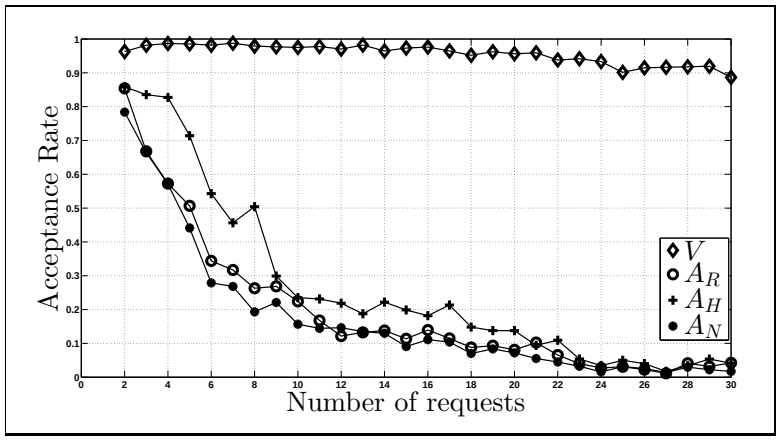

Figure 5.20: Acceptance rate for CLARKNET before reallocation

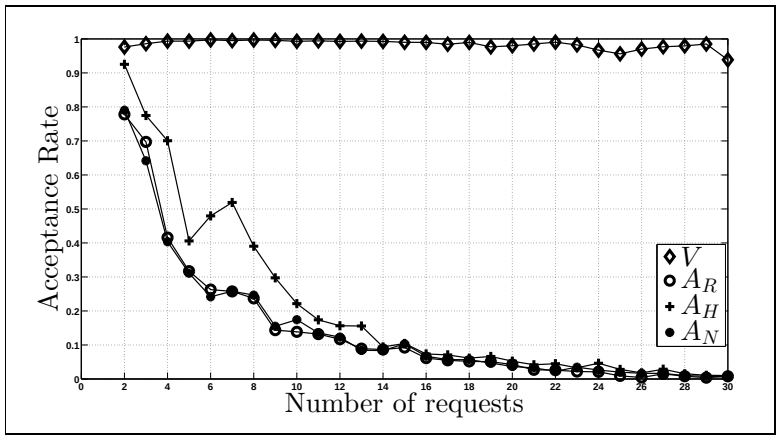

Figure 5.21: Acceptance rate for NASA before reallocation

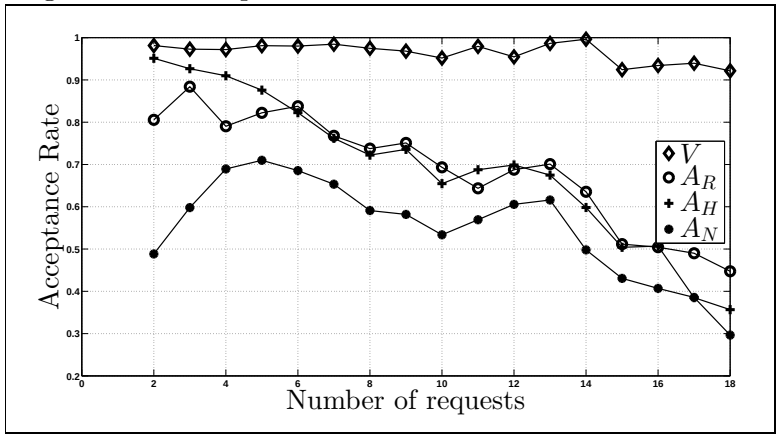

Figure 5.22: Acceptance rate for ESHOP before reallocation

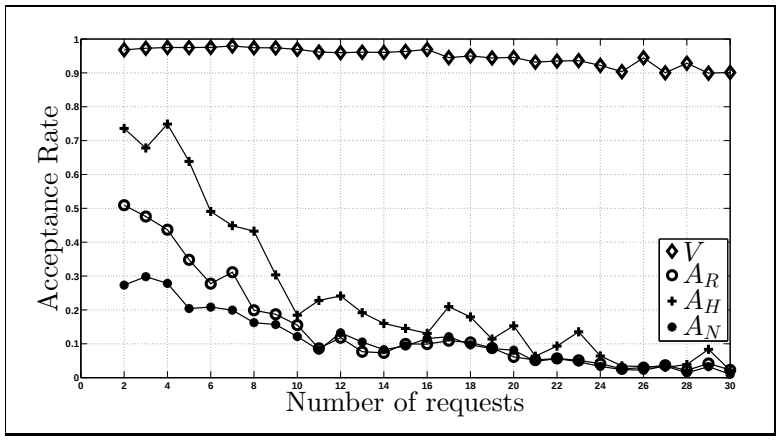

Figure 5.23: Acceptance rate for CLARKNET after reallocation

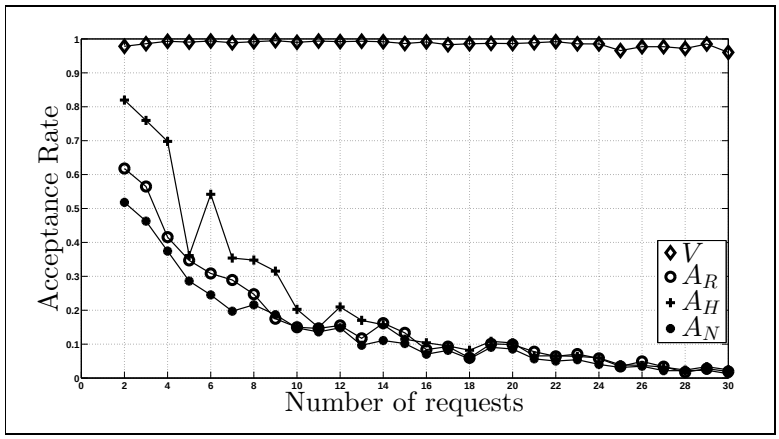

Figure 5.24: Acceptance rate for NASA after reallocation

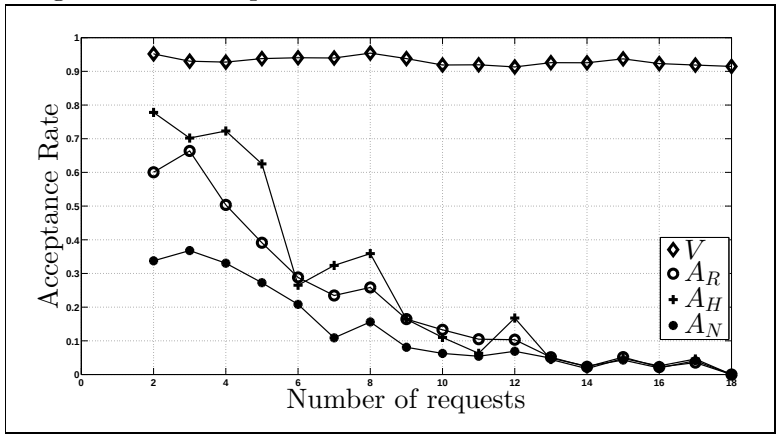

Figure 5.25: Acceptance rate for ESHOP after reallocation

5.5 Summary and Discussion

In this chapter, results of the methods described in the previous chapter have been provided. Initially, outcome of the clustering algorithm for MSNBC data set has been analysed. Afterwards, distribution of sequences across clusters has been presented. It has been shown that it is possible to detect on-going attack, by analysing sequence allocation to clusters. However, because legitimate sequences might be allocated to high-entropy clusters, and similarly, attacking sequences to low-entropy clusters, statistical methods are essential for intention classification. Subsequently, it has been shown that the methods applied for macro-interest are too restrictive. As a result it has been noted that improved classification can be achieved with extension of feature space. This has been confirmed with improved sequence distribution across clusters, and detection techniques for micro-interest approach.

Similarly, at the beginning, it has been presented that the distribution of sequences is very informative, and allows for attacking attempt detection. Moreover, legitimate sequence distributions are more similar for micro-interest approach, comparing to macro-interest approach. Subsequently, statistical analysis of sequences inside clusters is required. This is provided by application of (i) Bayes Factors and (ii) likelihood analysis. Both methods are able to detect variable strategies implemented in zombies. However, (i) require input from the attackers to perform detection. On the contrary, it has been shown that (ii) allows for scope-independent detection of attacking hosts. Specifically, whether the attacking traffic consists of multiple flooding hosts or rare crawlers, their browsing strategies have been detected. Moreover, reallocation of sequences improves detection of arriving hosts, especially for smaller data sets, in which data for training might be limited.

Chapter 6

Conclusion

In this thesis an attempt has been made to show that detection of different types of unsolicited web browsing can be achieved by application of clustering and likelihood analysis. Specifically, the term unsolicited web browsing refers to two types of malicious activity: (i) undesired crawling and (ii) flooding attacks. In general (i) refers to programmed machines that extracted sensitive information from websites; predominantly e-mail addresses. Moreover, it can also refer to site-mirroring applications that attempt to download entire website for copy-right infringement purposes. On the other hand, (ii) denotes unlawful usage of infected hosts, sending multiple GET requests, to exhaust resources at a web server. However, both, (i) and (ii) are performed in a legitimate way. Specifically, before running the attack, attackers attempt to discover existence of *rate-limiting* IDS, and select potentially *popular* web objects to use them during (ii). As a result, programmed machines, so-called zombies, request popular objects at legitimate rate, making their activity look as legitimate conduct. Moreover, observe that the difference between (i) and (ii) comes down to the application of web resources. Specifically, (i) defines malicious *usage* of website content. On the contrary, content is ignored in (ii), and only the *volume* of content possession (resulting in overloaded database or web server, due to the high rate of requests) is crucial.

As a result, a survey of current techniques has been provided to show that these methods underestimate the attacking strategies and their legitimate nature. Moreover, a data set with attacking traffic is unavailable. As a result, researchers end up generating and validating their approach on the manually obtained data set of attacking traffic. Specifically, there are two main groups of intrusion detection. Firstly, there are research approaches utilising sequence-dependent information, mainly, on transitions among web objects. However, it is shown that very often deviations of

the legitimate profile is defined as increased frequency of requested objects. On the other hand, there is research relying on protocol-related heuristic, and information encoded into Application layer protocol. This can vary from information on popular HTTP error codes to the ratio of different file types downloaded in a session. In addition, malicious behaviour violates HTTP heuristics by missing or invalid information in HTTP protocol packets or requests generating HTTP error codes.

However, it has been argued that to run attack successfully, attackers implement their zombies intelligently. In particular, it has been shown that there are existing tools for running either attack, and for the reference purpose, the copy of popular bot-net application has been analysed. The analysis of the code proves that legitimate nature of attacking hosts is easily implemented. Moreover, missing or invalid information of HTTP protocol specification can be straightforwardly set up. Therefore, one can argue that the only difference between illicit and valid web browsing is in *intention*. As a result, the measure of intention has been introduced, based on the similarity of interest in web objects. Specifically, it is assumed in research community that legitimate users visit similar web objects. Moreover, they should use similar pages to finally request the popular links. Therefore, a clustering algorithm is required to group sequences of web requests into classes expressing similar interests. In addition, the attacking strategies should be introduced to closely mimic legitimate patterns. As a result, the clustering algorithm cannot be used as a distinction between legitimate and attacking sequences. Specifically, because of similarity in behaviour, attacking sequences visit popular clusters of legitimate users. Moreover, legitimate users are assigned to clusters densely populated by attacking hosts. As a result, additional statistical measurement needs to be introduced to measure anomaly of sequences, on a per-cluster basis. This measure should be: attack-strategy independent (i.e. should be able to detect different browsing strategies chosen by attackers), and attack-scale independent (i.e. regardless of the number of browsing bots, the attempt should be discovered, especially when this number is very small, such as a lone crawler). Therefore, in the subsequent session the methods are summarised and their contribution to knowledge described.

6.1 Contribution to Knowledge

In this section the main contributions to knowledge are described. Specifically, these are: (i) the clustering algorithm, and (ii) the statistical measure that allows for detection of scale-independent traffic. Specifically, (i) should allow for *attempt* detection, and warns system administrators that there is an on-going attack. After-

wards, (ii) is introduced that allows for intention detection, regardless of the number of attacking hosts.

6.1.1 Clustering Algorithm

It is difficult to measure interest among categorical objects. Moreover, it is difficult to measure similarity of *data streams* in terms of data points appearing in the streams. However, if one could measure how "expected" a stream was, then it would be possible to explain validity of each sequence. Therefore, suppose that there is a composition of interest groups $C_{1:k}$ containing k clusters, for which each cluster $C_i, 1 \leq i \leq k$ corresponds to a particular interest group. Subsequently, suppose that there is a way to measure probability of a web sequence s inside the clusters $C_{1:k}$. The cluster, for which s attains the highest probability is chosen so that it describes the sequence s best.

Theoretical explanation of the algorithm, and sequence allocation have been described in Section 4.5 and Section 4.8. Specifically, web interest has been divided into two approaches: macro-interest and micro-interest approach. The former refers to interest in logical web categories, such as news, weather, for a news portal; or kitchen appliances and TV sets for an e-shop. On the other hand, micro-approach considers web objects *inside* the logical web categories (e.g. physical addresses of images, documents or other sub-categories inside a main category, say, weather).

Subsequently, the detailed results of the algorithm implementation have been described in Section 5.1 and Section 5.3.2. It has been shown that legitimate hosts frequently visit low-entropy clusters, and high-entropy clusters are sparsely populated with genuine hosts. On the other hand, the attacking hosts fail to discover the recent and popular pattern of transitions. As a result, the attacking hosts visit high-entropy clusters more often than the low-entropy ones. This phenomenon allows for detection of the on-going attack. If there is a reasonably large volume of attacking sequences, their distribution of requests across clusters differs from the training and validating of legitimate users. Observe that this is very important for further analysis. Essentially, the clustering algorithm does not take into account any time information, nor any protocol information is analysed. As a result, the interest in transitions among web objects allows for detection of the flooding attempt.

However, as it has been noted, the clustering algorithm does not provide clear distinction between attacking and legitimate users. As a result statistical measures are required to perform intention classification.

6.1.2 Statistical Measures

Initially, three statistical measures have been introduced for the macro-interest approach, based on: (i) Mahalanobis distance (see Section 4.11.1), (ii) mutual information (see Section 4.11.2), and (iii) likelihood of the longest same-category segment (see Section 4.11.3). In principle, (i) measures deviation from the expected profile of categorical requests. It performs very well against randomly-requesting zombies that frequently change a browsing category. Similarly, (ii) should discover unpopular transitions. Finally, (iii) has been introduced to detect hosts that remain inside one category for longer period of time.

In general, it has been shown that these statistical measurements provide satisfactory results against attacking sequences. However, their application turns out to be too restrictive against legitimate users. Thus, a focus has shifted from macro-interest into micro-interest. As a result, two statistical approaches have been introduced, and based on Bayes Factors and likelihood analysis. The former, assumed that there is a way to compare whether s is more similar to the legitimate profile of training sequences or resembles more the attacking sequences. It turns out that the application of Bayes Factors allows for such a discrimination, and the results of this operation have been presented in Section 5.4.1 and Section 5.4.2. Therefore, the obtained results show that intention classification is possible, and detection results are satisfactory. However, as it has been noted, the input of the attacking strategies is required to perform detection. Moreover, if the volume of attacking sequences is not large enough, then IDS is unable to perform intention classification correctly.

As a result, the method based on likelihood analysis has been introduced. This approach allows for intention classification, irrespective of the traffic volume and browsing strategies chosen by attackers. Specifically, because the input on the attacking strategies is not required, therefore, IDS is able to spot either a lone crawler deviating from the expected profile of browsing or flooding hosts with the same precision.

Furthermore, it has been observed that reallocation of sequences inside clusters allows for a better utilisation of likelihood-based measure. Specifically, once the clustering algorithm has been completed, there are many clusters that describe numerous interest groups. However, some of them are not very popular and should be merged with similar and more populated interest groups. As a result, the likelihood of observing legitimate users inside clusters increases. Moreover, the elimination of flat (i.e. unpopular) clusters, makes rare transitions encoded in attacking behaviour visible. The results of this analysis have been described in Section 5.4.3.

Chapter 7

Future Work

In general, there are two main areas this research could benefit from: (i) inclusion of temporal information and (ii) inter-cluster transition. Moreover, an improvement should be sought of (iii) initial number of clusters specification. Reallocation process shows that smaller cluster space could account for the same detection performance.

Essentially, recall that only popular links are selected for training the clustering algorithm (see Section 4.2 for details). This should be thought of as the first defence system against the Internet attackers, as this information is difficult to guess. As a result, (i) should be thought of as additional layer of defence. In principle, observe that time related information is not considered at this stage of the research. It is motivated by the research problem specification. In principle, it is assumed that initially attackers learn the minimum and maximum silence-period that is allowable by a rate-dependent IDS (see Section 3.2 for details). However, it could provide another level of defence against frequently-requesting zombies. In principle, the likelihood of sequences is conditioned on the previous requests. However, this could be extended by addition of time related information (i.e. silence-time information between two consecutive requests). Recall that likelihood of a sequence s_j is obtained as a product of transitions between two consecutive requests (see Section 4.8 for details). The extension of this approach could condition the probability of the next request not only on the previous, but also on the elapsed time between two requests. However, this increases space complexity of the system, and should be investigated further.

Subsequently, (ii) can be achieved with Hidden Markov Models. These models have been extensively used in intrusion detection domain, specifically the works of Lu et al.[52] and Xie et al. [99, 100, 101, 102] introduced the concept well. In principle, recall that every time a host makes a request its sequence of requests is

reassigned to a cluster for which the sequence attains highest likelihood of observation. As a result, as the sequence of requested web objects is observable information. However, transitions from one cluster to another can be perceived as a sequence of latent transitions - the transition from cluster to clusters remains hidden. As a result, the combination of (i) and (ii) should provide improvement to the current system of detection.

Bibliography

[1] Lada A. Adamic and Bernardo A. Huberman. Zipf's law and the Internet. *Glottometrics*, 3:143–150, 2002.

[2] Martin F. Arlitt and Carey L. Williamson. Web server workload characterization: the search for invariants. In *Proceedings of the 1996 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, SIGMETRICS '96, pages 126–137, New York, NY, USA, 1996. ACM. ISBN 0-89791-793-6. doi: 10.1145/233013.233034. URL `http://doi.acm.org/10.1145/233013.233034`.

[3] Daniel Barbará, Yi Li, and Julia Couto. Coolcat: an entropy-based algorithm for categorical clustering. In *Proceedings of the eleventh international conference on Information and knowledge management*, CIKM '02, pages 582–589, New York, NY, USA, 2002. ACM. ISBN 1-58113-492-4. doi: 10.1145/584792.584888. URL `http://doi.acm.org/10.1145/584792.584888`.

[4] Alessandro Basso and Stefano Sicco. Preventing massive automated access to web resources. *Computers & Security*, 28(34):174 – 188, 2009. ISSN 0167-4048. doi: http://dx.doi.org/10.1016/j.cose.2008.11.002. URL `http://www.sciencedirect.com/science/article/pii/S0167404808001156`.

[5] Andrew P. Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recogn.*, 30(7):1145–1159, July 1997. ISSN 0031-3203. doi: 10.1016/S0031-3203(96)00142-2. URL `http://dx.doi.org/10.1016/S0031-3203(96)00142-2`.

[6] Andrei Broder, Ravi Kumar, Farzin Maghoul, Prabhakar Raghavan, Sridhar Rajagopalan, Raymie Stata, Andrew Tomkins, and Janet Wiener. Graph structure in the web. *Comput. Netw.*, 33(1-6):309–320, June 2000. ISSN 1389-

1286. doi: 10.1016/S1389-1286(00)00083-9. URL http://dx.doi.org/10.1016/S1389-1286(00)00083-9.

[7] Christian Cachin and Jonathan A. Poritz. Secure intrusion-tolerant replication on the internet. In *Proceedings of the 2002 International Conference on Dependable Systems and Networks*, DSN '02, pages 167–176, Washington, DC, USA, 2002. IEEE Computer Society. ISBN 0-7695-1597-5. URL http://dl.acm.org/citation.cfm?id=647883.738262.

[8] R.K.C. Chang. Defending against flooding-based distributed denial-of-service attacks: a tutorial. *Communications Magazine, IEEE*, 40(10):42–51, 2002. ISSN 0163-6804. doi: 10.1109/MCOM.2002.1039856.

[9] Xin Chen and Xiaodong Zhang. A popularity-based prediction model for web prefetching. *Computer*, 36(3):63–70, March 2003. ISSN 0018-9162. doi: 10.1109/MC.2003.1185219. URL http://dx.doi.org/10.1109/MC.2003.1185219.

[10] Xuan Chen and John Heidemann. Flash crowd mitigation via adaptive admission control based on application-level observations. *ACM Trans. Internet Technol.*, 5(3):532–569, August 2005. ISSN 1533-5399. doi: 10.1145/1084772.1084776. URL http://doi.acm.org/10.1145/1084772.1084776.

[11] A. Corazza, S. Di Martino, F. Ferrucci, C. Gravino, and E. Mendes. Applying support vector regression for web effort estimation using a cross-company dataset. In *Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement*, ESEM '09, pages 191–202, Washington, DC, USA, 2009. IEEE Computer Society. ISBN 978-1-4244-4842-5. doi: 10.1109/ESEM.2009.5315991. URL http://dx.doi.org/10.1109/ESEM.2009.5315991.

[12] Bernstein Daniel Julius. Syn cookies, September 1996. URL http://cr.yp.to/syncookies.html.

[13] Dorothy E. Denning. An intrusion-detection model. *IEEE Trans. Softw. Eng.*, 13(2):222–232, February 1987. ISSN 0098-5589.

[14] Derek Doran and Swapna S. Gokhale. Web robot detection techniques: overview and limitations. *Data Min. Knowl. Discov.*, 22(1-2):183–210, January 2011. ISSN 1384-5810. doi: 10.1007/s10618-010-0180-z. URL http://dx.doi.org/10.1007/s10618-010-0180-z.

[15] Ehud Doron and Avishai Wool. Wda: A web farm distributed denial of service attack attenuator. *Comput. Netw.*, 55(5):1037–1051, April 2011. ISSN 1389-1286. doi: 10.1016/j.comnet.2010.05.001. URL `http://dx.doi.org/10.1016/j.comnet.2010.05.001`.

[16] Christos Douligeris and Aikaterini Mitrokotsa. Ddos attacks and defense mechanisms: classification and state-of-the-art. *Comput. Netw.*, 44(5):643–666, April 2004. ISSN 1389-1286. doi: 10.1016/j.comnet.2003.10.003. URL `http://dx.doi.org/10.1016/j.comnet.2003.10.003`.

[17] W. Eddy. RFC 4987:tcp syn flooding attacks and common mitigations, 2007.

[18] K. Egevang and P. Francis. RFC 1631: The IP network address translator (NAT), 1994.

[19] Karim El Defrawy, Minas Gjoka, and Athina Markopoulou. Bottorrent: misusing bittorrent to launch ddos attacks. In *Proceedings of the 3rd USENIX workshop on Steps to reducing unwanted traffic on the internet*, SRUTI'07, pages 1:1–1:6, Berkeley, CA, USA, 2007. USENIX Association. URL `http://dl.acm.org/citation.cfm?id=1361436.1361437`.

[20] D.M. Endres and J.E. Schindelin. A new metric for probability distributions. *Information Theory, IEEE Transactions on*, 49(7):1858–1860, 2003. ISSN 0018-9448.

[21] T. Fawcett. Roc graphs: Notes and practical considerations for researchers. *ReCALL*, 31(HPL-2003-4):1–38, 2004.

[22] P. Ferguson and D. Senie. RFC 2827:network ingress filtering: Defeating denial of service attacks which employ ip source address spoofing, 2000.

[23] Kevin Fu, Emil Sit, Kendra Smith, and Nick Feamster. Dos and don'ts of client authentication on the web. In *Proceedings of the 10th conference on USENIX Security Symposium - Volume 10*, SSYM'01, pages 19–19, Berkeley, CA, USA, 2001. USENIX Association. URL `http://dl.acm.org/citation.cfm?id=1267612.1267631`.

[24] D. Gavrilis, I. Chatzis, and E. Dermatas. Flash crowd detection using decoy hyperlinks. In *Networking, Sensing and Control, 2007 IEEE International Conference on*, pages 466–470, 2007. doi: 10.1109/ICNSC.2007.372823.

[25] Dimitris Gavrilis, Ioannis S. Chatzis, and Evangelos Dermatas. Detection of web denial-of-service attacks using decoy hyperlinks. In *Communication Systems, Networks Digital Signal Processing (CSNDSP), 2006 5th International Symposium on*, pages 266–270, 2006.

[26] Xianjun Geng and Andrew B. Whinston. Defeating distributed denial of service attacks. *IT Professional*, 2(4):36–41, July 2000. ISSN 1520-9202. doi: 10.1109/6294.869381. URL `http://dx.doi.org/10.1109/6294.869381`.

[27] Anup K. Ghosh, Aaron Schwartzbard, and Michael Schatz. Learning program behavior profiles for intrusion detection. In *Proceedings of the 1st conference on Workshop on Intrusion Detection and Network Monitoring - Volume 1*, ID'99, pages 6–6, Berkeley, CA, USA, 1999. USENIX Association. URL `http://dl.acm.org/citation.cfm?id=1267880.1267886`.

[28] Luis Campo Giralte, Cristina Conde, Isaac Martin de Diego, and Enrique Cabello. Detecting denial of service by modelling web-server behaviour. *Computers & Electrical Engineering*, In Press(0):–, 2012. ISSN 0045-7906. doi: http://dx.doi.org/10.1016/j.compeleceng.2012.07.004. URL `http://www.sciencedirect.com/science/article/pii/S0045790612001292`.

[29] Guofei Gu, Junjie Zhang, and Wenke Lee. BotSniffer: Detecting botnet command and control channels in network traffic. In *Proceedings of the 15th Annual Network and Distributed System Security Symposium (NDSS'08)*, February 2008.

[30] Christian Hennig. Cluster-wise assessment of cluster stability. *Computational Statistics and Data Analysis*, 52(1):258–271, September 2007. URL `http://ideas.repec.org/a/eee/csdana/v52y2007i1p258-271.html`.

[31] Gregoire Jacob, Engin Kirda, Christopher Kruegel, and Giovanni Vigna. Pubcrawl: protecting users and businesses from crawlers. In *Proceedings of the 21st USENIX conference on Security symposium*, Security'12, pages 25–25, Berkeley, CA, USA, 2012. USENIX Association. URL `http://dl.acm.org/citation.cfm?id=2362793.2362818`.

[32] Mehrdad Jalali, Norwati Mustapha, Md. Nasir Sulaiman, and Ali Mamat. Webpum: A web-based recommendation system to predict user future movements. *Expert Systems with Applications*, 37(9):6201 – 6212, 2010. ISSN 0957-4174. doi: 10.1016/j.eswa.2010.02.105. URL `http://www.sciencedirect.com/science/article/pii/S0957417410001442`.

[33] Liping Ji, Haijin Liang, Yitao Song, and XiaMu Niu. A normal-traffic network covert channel. In *Computational Intelligence and Security, 2009. CIS '09. International Conference on*, volume 1, pages 499–503, 2009. doi: 10.1109/CIS.2009.156.

[34] Evers Joris. Hacking for dollars, July 2005. URL `http://news.cnet.com/Hacking-for-dollars/2100-7349_3-5772238.htmll`.

[35] Jaeyeon Jung, Balachander Krishnamurthy, and Michael Rabinovich. Flash crowds and denial of service attacks: characterization and implications for cdns and web sites. In *Proceedings of the 11th international conference on World Wide Web*, WWW '02, pages 293–304, New York, NY, USA, 2002. ACM. ISBN 1-58113-449-5. doi: 10.1145/511446.511485. URL `http://doi.acm.org/10.1145/511446.511485`.

[36] Peyman Kabiri and Ali A. Ghorbani. Research on intrusion detection and response: A survey. *International Journal of Network Security*, 1:84–102, 2005.

[37] Georgios Kambourakis, Tassos Moschos, Dimitris Geneiatakis, and Stefanos Gritzalis. A fair solution to dns amplification attacks. In *Proceedings of the Second International Workshop on Digital Forensics and Incident Analysis*, WDFIA '07, pages 38–47, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-2941-0. doi: 10.1109/WDFIA.2007.2. URL `http://dx.doi.org/10.1109/WDFIA.2007.2`.

[38] Hongwen Kang, Kuansan Wang, David Soukal, Fritz Behr, and Zijian Zheng. Large-scale bot detection for search engines. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 501–510, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-799-8. doi: 10.1145/1772690.1772742. URL `http://doi.acm.org/10.1145/1772690.1772742`.

[39] Mehmed Kantardzic. *Data Mining: Concepts, Models, Methods and Algorithms*. John Wiley & Sons, Inc., New York, NY, USA, 2002. ISBN 0471228524.

[40] Robert E. Kass and Adrian E. Raftery. Bayes Factors. *Journal of the American Statistical Association*, 90(430):773–795, June 1995. ISSN 01621459. doi: 10.2307/2291091.

[41] S. Kullback and R. A. Leibler. On information and sufficiency. *Ann. Math. Statist.*, 22(1):79–86, 1951.

[42] Pradeep Kumar, P. Radha Krishna, Raju. S. Bapi, and Supriya Kumar De. Rough clustering of sequential data. *Data and Knowledge Engineering*, 63(2):183 – 199, 2007. ISSN 0169-023X. doi: 10.1016/j.datak.2007.01.003. URL http://www.sciencedirect.com/science/article/pii/S0169023X07000055.

[43] Terran Lane and CarlaE. Brodley. An empirical study of two approaches to sequence learning for anomaly detection. *Machine Learning*, 51(1):73–107, 2003. ISSN 0885-6125. doi: 10.1023/A:1021830128811. URL http://dx.doi.org/10.1023/A%3A1021830128811.

[44] M. Lecandowsky and D. Winter. Distance between Sets. *Nature*, 234(5323):34–35, 1971.

[45] Sangjae Lee, Gisung Kim, and Sehun Kim. Sequence-order-independent network profiling for detecting application layer ddos attacks. *EURASIP Journal on Wireless Communications and Networking*, 2011(1):1–9, 2011.

[46] Wenke Lee and Salvatore J. Stolfo. A framework for constructing features and models for intrusion detection systems. *ACM Trans. Inf. Syst. Secur.*, 3(4):227–261, November 2000. ISSN 1094-9224. doi: 10.1145/382912.382914. URL http://doi.acm.org/10.1145/382912.382914.

[47] Wenke Lee and Dong Xiang. Information-theoretic measures for anomaly detection. In *Proceedings of the 2001 IEEE Symposium on Security and Privacy*, SP '01, pages 130–, Washington, DC, USA, 2001. IEEE Computer Society. URL http://dl.acm.org/citation.cfm?id=882495.884435.

[48] Jonathan Lemon. Resisting syn flood dos attacks with a syn cache. In *Proceedings of the BSD Conference 2002 on BSD Conference*, BSDC'02, pages 10–10, Berkeley, CA, USA, 2002. USENIX Association. URL http://dl.acm.org/citation.cfm?id=1250894.1250904.

[49] Tao Li, Sheng Ma, and Mitsunori Ogihara. Entropy-based criterion in categorical clustering. In *Proceedings of the twenty-first international conference on Machine learning*, ICML '04, pages 68–, New York, NY, USA, 2004. ACM. ISBN 1-58113-838-5. doi: 10.1145/1015330.1015404. URL http://doi.acm.org/10.1145/1015330.1015404.

[50] Richard Lippmann, Joshua W. Haines, David J. Fried, Jonathan Korba, and Kumar Das. Analysis and results of the 1999 darpa off-line intrusion detection evaluation. In *Proceedings of the Third International Workshop on*

Recent Advances in Intrusion Detection, RAID '00, pages 162–182, London, UK, UK, 2000. Springer-Verlag. ISBN 3-540-41085-6. URL `http://dl.acm.org/citation.cfm?id=645838.670722`.

[51] Tom Lodewyckx, Woojae Kim, Michael D. Lee, Francis Tuerlinckx, Peter Kuppens, and Eric-Jan Wagenmakers. A tutorial on bayes factor estimation with the product space method. *Journal of Mathematical Psychology*, 55(5): 331 – 347, 2011. ISSN 0022-2496.

[52] Wei-Zhou Lu and Shun zheng Yu. An http flooding detection method based on browser behavior. In *Computational Intelligence and Security, 2006 International Conference on*, volume 2, pages 1151–1154, 2006. doi: 10.1109/ICCIAS.2006.295444.

[53] Wei-Zhou Lu and Shun zheng Yu. Web robot detection based on hidden markov model. In *Communications, Circuits and Systems Proceedings, 2006 International Conference on*, volume 3, pages 1806–1810, 2006. doi: 10.1109/ICCCAS.2006.285024.

[54] T.F. Lunt and R. Jagannathan. A prototype real-time intrusion-detection expert system. In *Security and Privacy, 1988. Proceedings., 1988 IEEE Symposium on*, pages 59–66, 1988. doi: 10.1109/SECPRI.1988.8098.

[55] Matthew V. Mahoney and Philip K. Chan. An analysis of the 1999 darpa/lincoln laboratory evaluation data for network anomaly detection. In *In Proceedings of the Sixth International Symposium on Recent Advances in Intrusion Detection*, pages 220–237. Springer-Verlag, 2003.

[56] Jelena Mirkovic and Peter Reiher. A taxonomy of ddos attack and ddos defense mechanisms. *SIGCOMM Comput. Commun. Rev.*, 34(2):39–53, April 2004. ISSN 0146-4833. doi: 10.1145/997150.997156. URL `http://doi.acm.org/10.1145/997150.997156`.

[57] David Moore, Colleen Shannon, Douglas J. Brown, Geoffrey M. Voelker, and Stefan Savage. Inferring internet denial-of-service activity. *ACM Trans. Comput. Syst.*, 24(2):115–139, May 2006. ISSN 0734-2071. doi: 10.1145/1132026.1132027. URL `http://doi.acm.org/10.1145/1132026.1132027`.

[58] Greg Mori and Jitendra Malik. Recognizing objects in adversarial clutter: breaking a visual captcha. In *Proceedings of the 2003 IEEE computer society conference on Computer vision and pattern recognition*, CVPR'03, pages

134–141, Washington, DC, USA, 2003. IEEE Computer Society. ISBN 0-7695-1900-8, 978-0-7695-1900-5. URL http://dl.acm.org/citation.cfm?id=1965841.1965858.

[59] Thi Thanh Sang Nguyen, Hai-Yan Lu, and Jie Lu. Ontology-style web usage model for semantic web applications. In *Intelligent Systems Design and Applications (ISDA), 2010 10th International Conference on*, pages 784–789, 2010. doi: 10.1109/ISDA.2010.5687169.

[60] Thi Thanh Sang Nguyen, Hai-Yan Lu, and Jie Lu. Ontology-style web usage model for semantic web applications. In *Intelligent Systems Design and Applications (ISDA), 2010 10th International Conference on*, pages 784–789, 2010. doi: 10.1109/ISDA.2010.5687169.

[61] L. Niven. *All the Myriad Ways*. Ballantine Books, 1971. ISBN 0-345-24084-7.

[62] G. Oikonomou and J. Mirkovic. Modeling human behavior for defense against flash-crowd attacks. In *Communications, 2009. ICC '09. IEEE International Conference on*, pages 1–6, 2009. doi: 10.1109/ICC.2009.5199191.

[63] Ruoming Pang, Vinod Yegneswaran, Paul Barford, Vern Paxson, and Larry Peterson. Characteristics of internet background radiation. In *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, IMC '04, pages 27–40, New York, NY, USA, 2004. ACM. ISBN 1-58113-821-0. doi: 10.1145/1028788.1028794. URL http://doi.acm.org/10.1145/1028788.1028794.

[64] KyoungSoo Park, Vivek S. Pai, Kang-Won Lee, and Seraphin Calo. Securing web service by automatic robot detection. In *Proceedings of the annual conference on USENIX '06 Annual Technical Conference*, ATEC '06, pages 23–23, Berkeley, CA, USA, 2006. USENIX Association. URL http://dl.acm.org/citation.cfm?id=1267359.1267382.

[65] Vern Paxson. An analysis of using reflectors for distributed denial-of-service attacks. *SIGCOMM Comput. Commun. Rev.*, 31(3):38–47, July 2001. ISSN 0146-4833. doi: 10.1145/505659.505664. URL http://doi.acm.org/10.1145/505659.505664.

[66] Vern Paxson and Sally Floyd. Wide area traffic: the failure of poisson modeling. *IEEE/ACM Trans. Netw.*, 3(3):226–244, June 1995. ISSN 1063-6692. doi: 10.1109/90.392383. URL http://dx.doi.org/10.1109/90.392383.

[67] James Pitkow. In search of reliable usage data on the www. *Comput. Netw. ISDN Syst.*, 29(8-13):1343–1355, September 1997. ISSN 0169-7552. doi: 10.1016/S0169-7552(97)00021-4. URL `http://dx.doi.org/10.1016/S0169-7552(97)00021-4`.

[68] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989. ISSN 0018-9219.

[69] Moheeb Abu Rajab, Jay Zarfoss, Fabian Monrose, and Andreas Terzis. My botnet is bigger than yours (maybe, better than yours): why size estimates remain challenging. In *Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, HotBots'07, pages 5–5, Berkeley, CA, USA, 2007. USENIX Association. URL `http://dl.acm.org/citation.cfm?id=1323128.1323133`.

[70] S. Ranjan, R. Swaminathan, M. Uysal, and E. Knightly. Ddos-resilient scheduling to counter application layer attacks under imperfect detection. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–13, 2006. doi: 10.1109/INFOCOM.2006.127.

[71] Supranamaya Ranjan. *High performance distributed denial-of-service resilient web cluster architecture*. PhD thesis, William Marsh Rice University, Houston, TX, USA, 2006. AAI3216765.

[72] Supranamaya Ranjan, Ram Swaminathan, Mustafa Uysal, Antonio Nucci, and Edward Knightly. Ddos-shield: Ddos-resilient scheduling to counter application layer attacks. *IEEE/ACM Trans. Netw.*, 17(1):26–39, February 2009. ISSN 1063-6692. doi: 10.1109/TNET.2008.926503. URL `http://dx.doi.org/10.1109/TNET.2008.926503`.

[73] Martin Roesch. Snort - lightweight intrusion detection for networks. In *Proceedings of the 13th USENIX conference on System administration*, LISA '99, pages 229–238, Berkeley, CA, USA, 1999. USENIX Association. URL `http://dl.acm.org/citation.cfm?id=1039834.1039864`.

[74] Robert E. Schapire. A brief introduction to boosting. In *Proceedings of the 16th international joint conference on Artificial intelligence - Volume 2*, IJCAI'99, pages 1401–1406, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc. URL `http://dl.acm.org/citation.cfm?id=1624312.1624417`.

[75] C.L. Schuba, I.V. Krsul, M.G. Kuhn, E.H. Spafford, A. Sundaram, and D. Zamboni. Analysis of a denial of service attack on tcp. In *Security and Privacy, 1997. Proceedings., 1997 IEEE Symposium on*, pages 208–223, 1997. doi: 10.1109/SECPRI.1997.601338.

[76] C. E. Shannon. A mathematical theory of communication. *Bell system technical journal*, 27, 1948.

[77] Michel Speiser, Gianluca Antonini, Abderrahim Labbi, and Juliana Sutanto. On nested palindromes in clickstream data. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '12, pages 1460–1468, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1462-6. doi: 10.1145/2339530.2339758. URL `http://doi.acm.org/10.1145/2339530.2339758`.

[78] M. Srivatsa, A. Iyengar, Jian Yin, and Ling Liu. A client-transparent approach to defend against denial of service attacks. In *Reliable Distributed Systems, 2006. SRDS '06. 25th IEEE Symposium on*, pages 61–70, 2006. doi: 10.1109/SRDS.2006.6.

[79] Mudhakar Srivatsa, Arun Iyengar, Jian Yin, and Ling Liu. A middleware system for protecting against application level denial of service attacks. In *Proceedings of the ACM/IFIP/USENIX 2006 International Conference on Middleware*, Middleware '06, pages 260–280, New York, NY, USA, 2006. Springer-Verlag New York, Inc. URL `http://dl.acm.org/citation.cfm?id=1515984.1516005`.

[80] Mudhakar Srivatsa, Arun Iyengar, Jian Yin, and Ling Liu. Mitigating application-level denial of service attacks on web servers: A client-transparent approach. *ACM Trans. Web*, 2(3):15:1–15:49, July 2008. ISSN 1559-1131. doi: 10.1145/1377488.1377489. URL `http://doi.acm.org/10.1145/1377488.1377489`.

[81] Dusan Stevanovic, Aijun An, and Natalija Vlajic. Detecting web crawlers from web server access logs with data mining classifiers. In *Proceedings of the 19th international conference on Foundations of intelligent systems*, ISMIS'11, pages 483–489, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-21915-3. URL `http://dl.acm.org/citation.cfm?id=2029759.2029822`.

[82] Dusan Stevanovic, Natalija Vlajic, and Aijun An. Unsupervised clustering of web sessions to detect malicious and non-malicious website users. *Proce-*

dia Computer Science, 5(0):123 – 131, 2011. ISSN 1877-0509. doi: http://dx.doi.org/10.1016/j.procs.2011.07.018. URL http://www.sciencedirect.com/science/article/pii/S1877050911003437. ¡ce:title¿The 2nd International Conference on Ambient Systems, Networks and Technologies (ANT-2011) / The 8th International Conference on Mobile Web Information Systems (MobiWIS 2011)¡/ce:title¿.

[83] Dusan Stevanovic, Aijun An, and Natalija Vlajic. Feature evaluation for web crawler detection with data mining techniques. *Expert Syst. Appl.*, 39(10): 8707–8717, August 2012. ISSN 0957-4174. doi: 10.1016/j.eswa.2012.01.210. URL http://dx.doi.org/10.1016/j.eswa.2012.01.210.

[84] Dusan Stevanovic, Natalija Vlajic, and Aijun An. Detection of malicious and non-malicious website visitors using unsupervised neural network learning. *Appl. Soft Comput.*, 13(1):698–708, January 2013. ISSN 1568-4946. doi: 10.1016/j.asoc.2012.08.028. URL http://dx.doi.org/10.1016/j.asoc.2012.08.028.

[85] Brett Stone-Gross, Marco Cova, Lorenzo Cavallaro, Bob Gilbert, Martin Szydlowski, Richard Kemmerer, Christopher Kruegel, and Giovanni Vigna. Your botnet is my botnet: analysis of a botnet takeover. In *Proceedings of the 16th ACM conference on Computer and communications security*, CCS '09, pages 635–647, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-894-0. doi: 10.1145/1653662.1653738. URL http://doi.acm.org/10.1145/1653662.1653738.

[86] Arijit Sur, AnandS. Nair, Abhishek Kumar, Apul Jain, and Sukumar Nandi. Steganalysis of network packet length based data hiding. *Circuits, Systems, and Signal Processing*, pages 1–18, 2012. ISSN 0278-081X. doi: 10.1007/s00034-012-9497-8. URL http://dx.doi.org/10.1007/s00034-012-9497-8.

[87] Arijit Sur, AnandS. Nair, Abhishek Kumar, Apul Jain, and Sukumar Nandi. Steganalysis of network packet length based data hiding. *Circuits, Systems, and Signal Processing*, pages 1–18, 2012. ISSN 0278-081X. doi: 10.1007/s00034-012-9497-8. URL http://dx.doi.org/10.1007/s00034-012-9497-8.

[88] Pang-Ning Tan and Vipin Kumar. Modeling of web robot navigational patterns, 2000.

[89] Pang-Ning Tan and Vipin Kumar. Discovery of web robot sessions based on their navigational patterns. *Data Min. Knowl. Discov.*, 6(1):9–35, January 2002. ISSN 1384-5810. doi: 10.1023/A:1013228602957. URL `http://dx.doi.org/10.1023/A:1013228602957`.

[90] Andrew Tanenbaum. *Computer Networks*. Prentice Hall Professional Technical Reference, 4th edition, 2002. ISBN 0130661023.

[91] T. Thapngam, Shui Yu, Wanlei Zhou, and G. Beliakov. Discriminating ddos attack traffic from flash crowd through packet arrival patterns. In *Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on*, pages 952–957, 2011. doi: 10.1109/INFCOMW.2011.5928950.

[92] G. Vigna and R. A. Kemmerer. Netstat: A network-based intrusion detection approach. In *Proceedings of the 14th Annual Computer Security Applications Conference*, ACSAC '98, pages 25–, Washington, DC, USA, 1998. IEEE Computer Society. ISBN 0-8186-8789-4. URL `http://dl.acm.org/citation.cfm?id=784589.784632`.

[93] K.K.K. Wan and R.K.C. Chang. Engineering of a global defense infrastructure for ddos attacks. In *Networks, 2002. ICON 2002. 10th IEEE International Conference on*, pages 419–427, 2002. doi: 10.1109/ICON.2002.1033348.

[94] Jin Wang, Xiaolong Yang, and Keping Long. A new relative entropy based app-ddos detection method. In *Computers and Communications (ISCC), 2010 IEEE Symposium on*, pages 966–968, 2010. doi: 10.1109/ISCC.2010.5546587.

[95] Jin Wang, Xiaolong Yang, and Keping Long. Web ddos detection schemes based on measuring user's access behavior with large deviation. In *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, pages 1–5, 2011. doi: 10.1109/GLOCOM.2011.6133798.

[96] N. Weiler. Honeypots for distributed denial-of-service attacks. In *Enabling Technologies: Infrastructure for Collaborative Enterprises, 2002. WET ICE 2002. Proceedings. Eleventh IEEE International Workshops on*, pages 109–114, 2002. doi: 10.1109/ENABL.2002.1029997.

[97] Patrick Wendell and Michael J. Freedman. Going viral: flash crowds in an open cdn. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, IMC '11, pages 549–558, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-1013-0. doi: 10.1145/2068816.2068867. URL `http://doi.acm.org/10.1145/2068816.2068867`.

[98] Y. Xie, S. Tang, X. Huang, C. Tang, and X. Liu. Detecting latent attack behavior from aggregated web traffic. *Comput. Commun.*, 36(8):895–907, May 2013. ISSN 0140-3664. doi: 10.1016/j.comcom.2013.01.013. URL `http://dx.doi.org/10.1016/j.comcom.2013.01.013`.

[99] Yi Xie and Xiangnong Huang. Http-session model and its application in anomaly http traffic detection. In *Proceedings of the 2010 Sixth International Conference on Semantics, Knowledge and Grids*, SKG '10, pages 141–148, Washington, DC, USA, 2010. IEEE Computer Society. ISBN 978-0-7695-4189-1. doi: 10.1109/SKG.2010.24. URL `http://dx.doi.org/10.1109/SKG.2010.24`.

[100] Yi Xie and Shun-Zheng Yu. A large-scale hidden semi-markov model for anomaly detection on user browsing behaviors. *IEEE/ACM Trans. Netw.*, 17 (1):54–65, February 2009. ISSN 1063-6692. doi: 10.1109/TNET.2008.923716. URL `http://dx.doi.org/10.1109/TNET.2008.923716`.

[101] Yi Xie and Shun-Zheng Yu. Monitoring the application-layer ddos attacks for popular websites. *IEEE/ACM Trans. Netw.*, 17(1):15–25, February 2009. ISSN 1063-6692. doi: 10.1109/TNET.2008.925628. URL `http://dx.doi.org/10.1109/TNET.2008.925628`.

[102] Yi Xie and Shun zheng Yu. A novel model for detecting application layer ddos attacks. In *Computer and Computational Sciences, 2006. IMSCCS '06. First International Multi-Symposiums on*, volume 2, pages 56–63, 2006. doi: 10.1109/IMSCCS.2006.159.

[103] Yi Xie, S. Tang, Y. Xiang, and J. Hu. Resisting web proxy-based http attacks by temporal and spatial locality behavior. *IEEE Transactions on Parallel and Distributed Systems*, 24(7):1401–1410, 2013. ISSN 1045-9219. doi: http://doi.ieeecomputersociety.org/10.1109/TPDS.2012.232.

[104] J. Yan and A.S. El Ahmad. Breaking visual captchas with naive pattern recognition algorithms. In *Computer Security Applications Conference, 2007. ACSAC 2007. Twenty-Third Annual*, pages 279–291, 2007. doi: 10.1109/ACSAC.2007.47.

[105] Qiang Yang, Joshua Zhexue Huang, and Michael Ng. A data cube model for prediction-based web prefetching. *J. Intell. Inf. Syst.*, 20(1):11–30, January 2003. ISSN 0925-9902. doi: 10.1023/A:1020990805004. URL `http://dx.doi.org/10.1023/A:1020990805004`.

[106] T. Yatagai, T. Isohara, and Iwao Sasase. Detection of http-get flood attack based on analysis of page access behavior. In *Communications, Computers and Signal Processing, 2007. PacRim 2007. IEEE Pacific Rim Conference on*, pages 232–235, 2007. doi: 10.1109/PACRIM.2007.4313218.

[107] Nong Ye, Syed Masum Emran, Qiang Chen, and Sean Vilbert. Multivariate statistical analysis of audit trails for host-based intrusion detection. *IEEE Trans. Comput.*, 51(7):810–820, July 2002. ISSN 0018-9340. doi: 10.1109/TC.2002.1017701. URL http://dx.doi.org/10.1109/TC.2002.1017701.

[108] Dit-Yan Yeung and Yuxin Ding. Host-based intrusion detection using dynamic and static behavioral models. *Pattern Recognition*, 36(1):229 – 243, 2003. ISSN 0031-3203. doi: http://dx.doi.org/10.1016/S0031-3203(02)00026-2. URL http://www.sciencedirect.com/science/article/pii/S0031320302000262.

[109] Jie Yu, Chengfang Fang, Liming Lu, and Zhoujun Li. A lightweight mechanism to mitigate application layer ddos attacks. In Peter Mueller, Jian-Nong Cao, and Cho-Li Wang, editors, *Scalable Information Systems*, volume 18 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 175–191. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-10484-8. doi: 10.1007/978-3-642-10485-5_13. URL http://dx.doi.org/10.1007/978-3-642-10485-5_13.

[110] Shui Yu, Wanlei Zhou, and R. Doss. Information theory based detection against network behavior mimicking ddos attacks. *Communications Letters, IEEE*, 12(4):318–321, 2008. ISSN 1089-7798. doi: 10.1109/LCOMM.2008.072049.

[111] Shui Yu, Theerasak Thapngam, Jianwen Liu, Su Wei, and Wanlei Zhou. Discriminating ddos flows from flash crowds using information distance. In *Proceedings of the 2009 Third International Conference on Network and System Security*, NSS '09, pages 351–356, Washington, DC, USA, 2009. IEEE Computer Society. ISBN 978-0-7695-3838-9. doi: 10.1109/NSS.2009.29. URL http://dx.doi.org/10.1109/NSS.2009.29.

[112] Shui Yu, Guofeng Zhao, Song Guo, Yang Xiang, and A.V. Vasilakos. Browsing behavior mimicking attacks on popular web sites for large botnets. In *Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on*, pages 947–951, 2011. doi: 10.1109/INFCOMW.2011.5928949.

[113] Shui Yu, Song Guo, and I. Stojmenovic. Can we beat legitimate cyber behavior mimicking attacks from botnets? In *INFOCOM, 2012 Proceedings IEEE*, pages 2851–2855, 2012. doi: 10.1109/INFCOM.2012.6195714.

[114] Shui Yu, Guofeng Zhao, Wanchun Dou, and S. James. Predicted packet padding for anonymous web browsing against traffic analysis attacks. *Information Forensics and Security, IEEE Transactions on*, 7(4):1381–1393, 2012. ISSN 1556-6013. doi: 10.1109/TIFS.2012.2197392.

[115] Shui Yu, Wanlei Zhou, Wanchun Dou, and S.K. Makki. Why it is hard to fight against cyber criminals? In *Distributed Computing Systems Workshops (ICDCSW), 2012 32nd International Conference on*, pages 537–541, 2012. doi: 10.1109/ICDCSW.2012.25.

[116] Shui Yu, Wanlei Zhou, Weijia Jia, Song Guo, Yong Xiang, and Feilong Tang. Discriminating ddos attacks from flash crowds using flow correlation coefficient. *IEEE Transactions on Parallel and Distributed Systems*, 23(6):1073–1080, 2012. ISSN 1045-9219. doi: http://doi.ieeecomputersociety.org/10.1109/TPDS.2011.262.

[117] Shui Yu, Wanlei Zhou, Weijia Jia, and Jiankun Hu. Attacking anonymous web browsing at local area networks through browsing dynamics. *Comput. J.*, 55(4):410–421, April 2012. ISSN 0010-4620. doi: 10.1093/comjnl/bxr065. URL `http://dx.doi.org/10.1093/comjnl/bxr065`.

[118] Stefano Zanero. Behavioral intrusion detection. In Cevdet Aykanat, Turul Dayar, and brahim Krpeolu, editors, *Computer and Information Sciences - ISCIS 2004*, volume 3280 of *Lecture Notes in Computer Science*, pages 657–666. Springer Berlin Heidelberg, 2004. ISBN 978-3-540-23526-2. doi: 10.1007/978-3-540-30182-0_66. URL `http://dx.doi.org/10.1007/978-3-540-30182-0_66`.

[119] Yongguang Zhang, Wenke Lee, and Yi-An Huang. Intrusion detection techniques for mobile wireless networks. *Wirel. Netw.*, 9(5):545–556, September 2003. ISSN 1022-0038. doi: 10.1023/A:1024600519144. URL `http://dx.doi.org/10.1023/A:1024600519144`.

[120] Jianwei Zhuge, Thorsten Holz, Xinhui Han, Jinpeng Guo, and Wei Zou. Characterizing the IRC-based Botnet Phenomenon. Technical report, Reihne Informatik, December 2007.